SLIDING WINDOWS AND CNNS FOR AUTO-HARMONIZATION

Aldo Aguilar, Alex Reneau, Barry Zhang

Northwestern University

1. MOTIVATION

Harmonization is a crucial part of what makes music enjoyable. The ability to generate harmonies requires a very deep understanding of the underlying structures of music pieces. A deep learning model able to learn to harmonize melodies would then give us great insights into how these models learn the structures of music, which can be instrumental to other musical-audio settings.

We would like to build an automatic melody harmonizer from lead sheet music that produces simple chordal accompaniment (in the forms of first inversion triads) for a symbolic melody. We would like to assess this both quantitatively, by checking against the original chord progression, and qualitatively, by assessing whether the chord progression is enjoyable or appropriate for the given melody.

2. PRIOR WORK

In Chord Generation From Symbolic Melody Using BLSTM Networks, Lim et al. used a bidirectional LSTM to accomplish the task and was able to reach higher accuracy (around 50%) than previous HMM and DNN-HMM based approaches. Yamada et. al. also utilized a recurrent neural network to delve into the issue of four-part harmonization by comparing it to a Bayesian network. The comparison found that recurrent neural network output dull alto and tenor lines but smoother bass lines.

3. DATASET

The dataset we are using is from Lim et al [1]. by the Music Audio Research Group. The dataset consists of 2252 Western songs from many different genres. They are stored in lead sheet formats like shown in the image above. For our purposes, the only features we are considering are the key_mode and the note_root. We are encoding both the nodes and the chords using a simple dictionary such that we have 27 unique integers for both notes and chords: 2 keys (major/minor) * 13 semitones (12 semitones + rest) + 1 padding token. We decided to include the major/minor information in our notes as well for more information about the tone of the piece.

4. SLIDING WINDOW METHOD

Motivated by [2] and [3], we decided to implement a sliding window method for this project as a means to capture temporal dependencies within songs. A sliding window method is a common method in machine learning that is used to capture temporal dependencies in the data. It enforces that each time step includes a specified number of previous time steps in the current decision. Using a sliding window method has proven to be useful in human-robot interaction, stock forecasting, and data mining [3] [4] [5].

Our task works under the assumption that there are temporal dependencies for each song. For our work, the purpose of implementing a sliding window method enabled us to use a convolutional neural network or standard machine learning algorithms [1] [2]. Of course, there are other ways to capture temporal dependencies using deep learning (i.e. recurrent nets), but we found the sliding window method to be more useful for this task because it provides us with better results.

Our implementation of the sliding window method is unique in the sense that it does not flatten the current and specified previous feature vectors into one feature vector. Instead, we package the current feature vector and the specified number of previous feature vectors into a matrix. The motivation for building this input representation for each time step was that it would provide a data representation that was optimal to feed into a convolutional neural network.

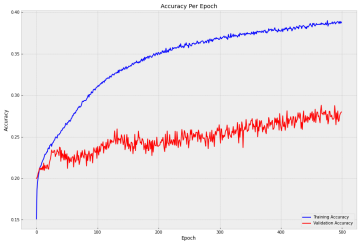
$$\hat{X}_t = [X_t, X_{t-1}, X_{t-2}, ..., X_{t-(s-1)}]$$

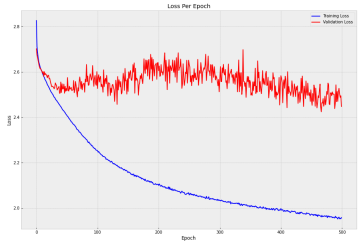
X represents all of the measures, t is the current time step, and s is the window size parameter that determines how many previous measures the current matrix, X_t, contains, or the number of measures per matrix. X represents our new set of data with altered representations for each time step. Without implementing the sliding window method, each feature vector has a window size of 1 because the length of a window is equal to the length of each measure. X_t is a measure in X. For each X_t, we package X_t and the $s-1$ previous measures into matrix X_t. If $t < s$, we pad X_t with zero vectors to account for the missing measures. After we have applied our sliding window method to each X_t, we obtain the windowed data, X_t, that is then passed into the convolutional network for training, validating, and testing.

5. TRAINING AND TESTING

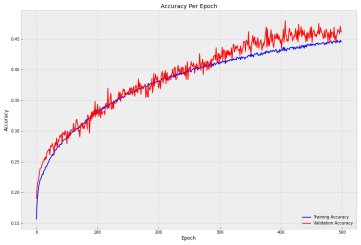
We train a three-layer convolutional neural network. It has two convolutional layers and a relu activation function after each convolutional layer and a final linear layer. Our model also uses 0.2 dropout to help with overfitting. We train our model for 500 epochs and use a batch size of 512 (same as Lim et al.). The model was trained using the Adam optimizer with a learning rate of .001. Cross entropy was used as our loss function, we ignored padding to prevent the model from outputting invalid predictions/chords. We do a 80/20 split of the original training set to achieve our training and validation sets. We test our model on a held-out test set. We used the accuracy metric to assess the quality of our model's performance. We retrain our model for every window size 1-30 to assess the impact of the window size. The training time ranged from seven minutes to approximately a half an hour while training on a Nvidia 2080 Ti. Here is the Accuracy and loss graphs of window sizes 1, 2 and 4.

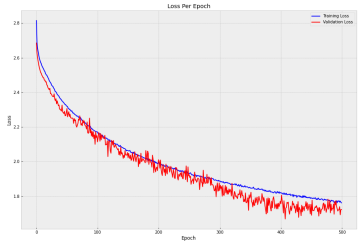
Window size 1



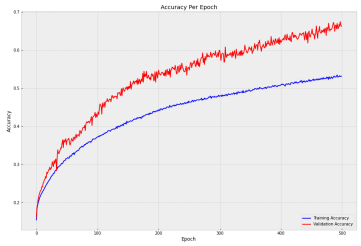


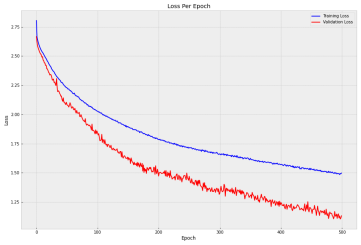
Window size 2





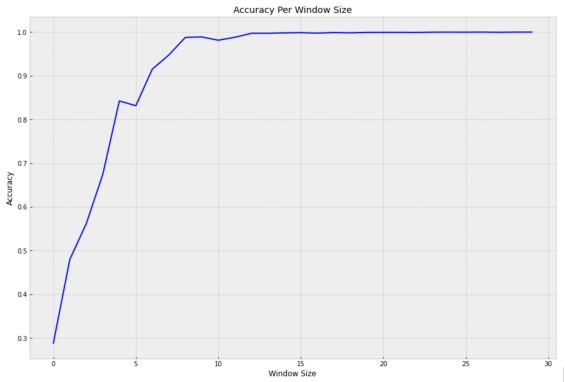
Window size 4





6. RESULTS

Utilizing the sliding window method with a convolutional neural network our model was able to achieve a very high accuracy of $> 90\%$ predictions for chords on the Music Audio Research Group's dataset. We tested a variety of models which utilized different architectures adjusted for each size of the sliding window. As we increased the size of the sliding window we found that the model's performance would converge much quicker. It seems that by including more temporal information the model is able to learn relationships in the data much faster.



Above is a figure which shows top accuracy achieved on the held out data from our dataset. We see at a window size of 0 we are just feeding in the current time step, the model performs better than random chance. Over time we see that the model's accuracy on the data plateaus around 99% accuracy. The plateau is achieved at a window size of 9.

7. DISCUSSION

Model Details

We used a modified version of the LeNet5 neural network. Our network has 2 convolutional layers which feed into 2 linear layers. All layers use the relu activation function. There is no pooling used in our network. Dropout is performed after each layer with a probability of .2. We did notice that during training our models performance was under that of the validation set. This is because of the dropout during training. The architecture utilizes same-paddings across the 3rd dimension of the input. By utilizing the same padding here we make it efficient for creating models which can handle different sliding window sizes.

Sliding Window

From our results we can see that the sliding window method has significant effects on our results. It provides us with better results and seems to show that as the window size increases, our performance increases. It is notable that after window size 15 our performance in accuracy becomes less important. This points out that our model less effectively captures temporal dependencies after window size 15.

One of the possible limitations of our sliding window method is that our window size is static. For our problem, this means that some time steps include information from other songs. This can be problematic because we want the model to focus on the temporal dependencies within each song, so for future work we might consider introducing a dynamic sliding window method.

Beat Original Paper's Performance

Our results prove to improve significantly upon the results of Lim et al. Their top accuracy was 51% and our top accuracy was 0.99%. This shows our work to be a strong contribution to the auto-harmonization space. We hypothesize that [1]. did not achieve better results because the BLSTM may have been overkill or not the correct deep learning model for this problem. We ran into many issues with re-implementing the BLSTM model from Lim et al. because it had difficulties with our data representation. However, the unique data representation that we pass into the convolutional neural network allows the deep network to better learn the problem space. This shows the importance of picking the appropriate model for your final input representation.

Limitations of current data representation

Our current symbolic representation of the music is still fairly limited. For the input melodies, we are omitting the note length as well as octave information. For the output chord progressions, we are generating only first inversion triads without distinguishing between any parts of the harmony. These omissions were deliberate for the scope of the project but these are important elements of a symbolic representation of music. In order to show that the model is able to effectively learn the underlying pattern in music, we still need to include a more complete set of features.

8. CONCLUSION

Our model shows promising results on the dataset from the Music Audio Research Group. However, more testing is needed before we can verify the validity of our results. Moving forward we would like to test our model on predicting more complicated chords and including octaves, to verify that it is learning correctly. Regardless, the sliding window approach has shown promising results in learning temporal relationships in data.

9. REFERENCES

[1] Hyungui Lim, Seungyeon Rhyu, and Kyogu Lee, "Chord generation from symbolic melody using blstm networks," *arXiv preprint arXiv:1712.01011*, 2017.

[2] Thomas G Dietterich, "Machine learning for sequential data: A review," in *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*. Springer, 2002, pp. 15–30.

[3] Alex Reneau and Jason R. Wilson, "Supporting user autonomy with multimodal fusion to detect when a user needs assistance from a social robot," in *Proceedings of the AI-HRI Symposium at AAAI-FSS 2020*, 2020.

[4] HS Hota, Richa Handa, and AK Shrivas, "Time series data prediction using sliding window based rbf neural network," *International Journal of Computational Intelligence Research*, vol. 13, no. 5, pp. 1145–1156, 2017.

[5] Hua-Fu Li and Suh-Yin Lee, "Mining frequent itemsets over data streams using efficient window sliding techniques," *Expert systems with applications*, vol. 36, no. 2, pp. 1466–1477, 2009.