Chordinator

Fall 2023 - Audio Content Analysis - Final Project

Dani Leinwander Joseph Cleveland Dhruv Pargai Venkatakrishnan V K

I. Introduction

Currently a hot topic due to the recent uptick in machine learning research and applications, generative chord accompaniment models are available in abundance. Many different approaches have been implemented in the last few years - including but not limited to: training HMMs to realize figured bass in both real time and with pre-processing [1], the use of transformers to model multi track MIDI accompaniment [2], and comparing various kinds of generative progressions from GANs [3].

Given there are many ways to approach this question of generative accompaniment, in our final project we take the opportunity to experiment with building a pipeline that is able to take in raw monophonic audio and generate chord accompaniments. We experiment with using two models in this pipeline: a chord detection model (GRU) and a chord prediction model (CNN) which we call Chordinator. Github

II. Objectives

We found that in order to properly train a chord prediction model that generates chord accompaniment, we need to augment a currently available source separation dataset with chord labels. In order to accomplish this, we train a chord detection model (GRU) in order to augment the MusDB source separated dataset with chord labels at given time stamps.

- MusDB dataset augmentation
  - train chord detection model
  - MusDB dataset augmentation

We then train Chordinator, our chord prediction model (CNN) using our newly augmented dataset.

- train Chordinator (Chord Prediction Model)

We found that the output from the chord prediction model (Chordinator) was slightly noisy, with occasional small errors in chord outputs. Since our model predicts a chord at every six frames (about every .25 seconds), there are small deviations that appear due to random error. In order to handle this and provide users with clean output, we devised a post-processing algorithm to clean the output of those small deviations. We also create a web application in order to showcase our model's capabilities.

- post-processing algorithm
- web application

After finishing our pipeline, we received evaluation feedback from a volunteer who created raw audio and evaluated the resulting output. We use this evaluation to consider the successes and limitations of our model in the discussion.

- report accuracy metrics for each model
- human evaluation of final output
- detail successes and limitations

III. Methodology

A. Dataset Augmentation with Chord Labels

In order to train Chordinator, our chord prediction model, we need a dataset with (1) source separated melodies and (2) labeled chord changes. The MusDB dataset [4] contains close to 10 hours of source separated audio. However, the dataset does not contain labeled chords for training chord prediction models. In order to augment the MusDB dataset, we (1) train a chord detection model using our own Logic Chord Detection dataset (made using Logic pro) and (2) use the output to augment the MusDB dataset with chord labels.

1) **Chord Detection Model** Our Logic Chord Detection dataset contains 15 examples for each key and chord type (Major, Minor, Augmented, Diminished), 720 examples in total created using a variety of virtual instruments in Logic Pro X. A bi-directional gated recurrent unit (PyTorch GRU) is used as the detection model. The input to the model is a 12-dimensional chroma vector C_t, with a block size of 2048 and a hop of 256.
   - ***Train*** -
     The 12-dimensional output vector $c \in \mathbb{R}^{12}$ is passed through a softmax, and Cross Entropy Loss is computed with respect to the ground truth chord template $c_g \in \mathbb{R}^{12}$ and the model is trained using an Adam Optimizer. Since the $||c_g||_2 = \sqrt{3}$, we will normalize c_g to 1.
   - ***Inference*** -
     For inference, we will compute the cosine similarity between the output of GRU and a number of chord templates, similar to what is done by Oudre et al. [5]. Cosine similarity is defined as follows -

$$\cos(\theta) = \frac{A \cdot B}{||A||_2||B||_2} \tag{1}$$

     where $A, B \in \mathbb{R}^n$.
     Cosine similarity determines how close a certain n-dimensional vector is to another n-dimensional vector. Once the cosine similarity is computed for the output vector with every chord template, the

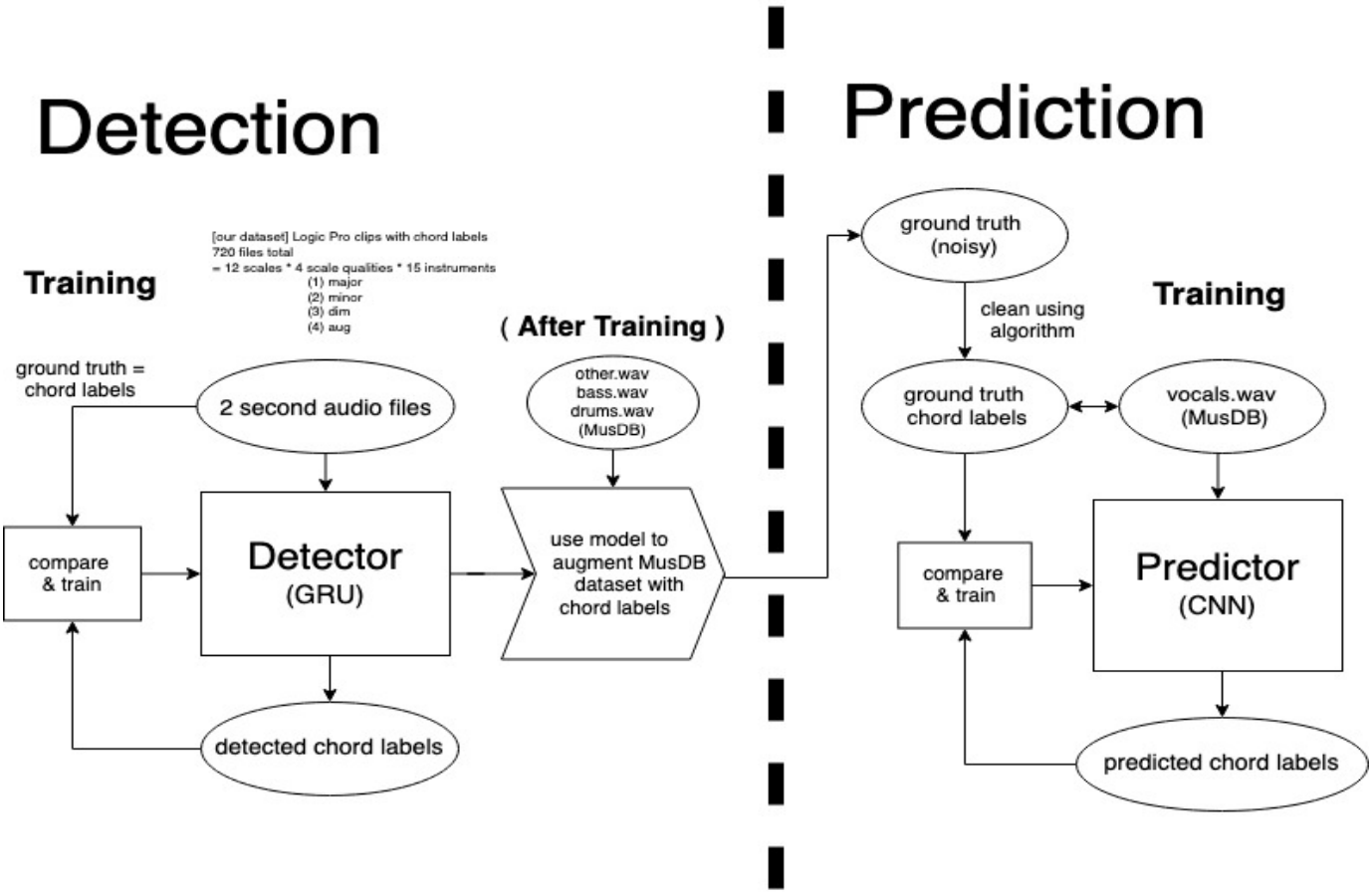

Fig. 1. Chordinator Pipeline

argmax of these values is outputted as the final chord.

$$\text{Chord Output} = \arg\max_i \frac{c \cdot c_{g_i}}{||c||_2 ||c_{g_i}||_2} \quad (2)$$

2) **MusDB Dataset Augmentation**
Using the trained chord detection model, we generate ground-truth labels for a chord prediction dataset. Each song from MuseDB is segmented into 6-block chunks. The accompaniment audio from `other.wav` is passed through the GRU to obtain a ground-truth chord label. This label $y \in \mathbb{R}^{12}$ is paired with the corresponding chunk of melody audio $X \in \mathbb{R}^{12\times 6}$ from `vocals.wav` to create a dataset of melody-chord pairs.

B. Chord Prediction

The Chordinator model is a 4-layer convolutional (Pytorch Conv2D) neural network with ReLU activation and batch normalization following each layer. A kernel size of (1, 3) is used for all layers. This essentially creates 1-dimensional convolutions across the time dimension with no dependence between adjacent chroma bins, whilst allowing the kernel to share weights for the processing of each bin. A final fully connected layer outputs a 12-dimensional vector, to which softmax is applied.

For the loss metric, Cross Entropy was used, as the network is free to output individual probabilities for each chroma. These are compared against the values of y which correspond to one of the chord templates from the data for the prior chord detection model. With sufficient training, the output of the final layer resembles a rigid 3-note chord template, despite the ability to choose unseen chords.

C. Post-Processing Algorithm

The output from our chord prediction model predicts a chord every six frames, or about every .25 seconds. The output mostly consists of the same chord being printed multiple times, however, due to unpredictable error such as noise in the signal, there is occasionally a chord that seems as though it does not belong.

In order to combat the noisy (unclean) output, we devise a rule based cleaning algorithm to omit those chords which appear to be the result of random error.

We first create an empty stack and consider chords, only if it has been continuously predicted more than 10 times, which is 60 chromagram frames. If so, the chord along with the first timestamp is pushed to the stack, else, it is discarded. This

Volunteer #1 Responses

					Satisfaction Ratings					
Song #	Song Name	Instrument	Key	Genre	Overall	Harmonic	Rhythmic	Correct Chord Count	Better/worse than original accompaniment to song	Additional comments
1	Giant Steps	flute	E flat Major	jazz	5	1	5	4	worse	The accompaniment of this major tune seems to maintain a minor quality throughout. The changes mostly matches up with the movement of the melody (rhythmic).
2	Giant Steps	flute 2	E flat Major	jazz	1	1	1	3	worse	The result of the first flute take was much better. The harmonic progression as a whole lacked coherence. The melody rarely lined up with the chords in a logical way (rhythm).
3	Giant Steps	vox	E flat Major	jazz	1	1	1	2	worse	Both the rhythmic and harmonic Both the rhythmic and harmonic feel of the original tune are lost in the AI accompaniment. Not enough movement (rhythmic)
4	500 Miles	alto saxophone	C minor	jazz	5	1	5	2	worse	The AI accompaniment has a strong major feel overall while the original tune is minor
5	500 Miles	soprano saxophone	C minor	jazz	5	1	5	4	worse	The soprano take was more successful than the alto take
6	500 Miles	flute	C minor	jazz	1	1	1	2	worse	AI accompaniment gave a very major feel to a minor tune
7	Solar	alto saxophone	C minor	jazz	5	5	1	4	worse	Lots of seemingly unnecessary chord movement for what is a relatively simple tune in reality
8	Solar	soprano saxophone	C minor	jazz	5	5	1	5	worse	The result of both the soprano and alto takes seemed to be about the same
Average Ratings out of 10					3.5	2	2.5			
Average Correct chords								3.25		

Fig. 2. Human Evaluation Responses

way, we eliminate the random noisy chords generated by our model.

IV. Results

1) Accuracy Metric - Chord Detection: For our chord detection model, we report a 98.44% validation accuracy rating.

2) Accuracy Metric - Chordinator (Chord Prediction): For our chord prediction model, we report a 66.04% validation accuracy rating.

3) Human Evaluation: Though we had limited time to evaluate these models (see discussion), we were able to get some human evaluation of our final pipeline through a volunteer. This volunteer made 8 recordings of monophonic audio of various instruments and songs (see Appendix). We then provided back to the volunteer the mixed audio of their input combined with audio MIDI chord representations of our Chordinator's output. The volunteer filled out an evaluation form with information about their musical experience, views on generative music from "AI" and questions about their satisfaction with each resulting output (see Appendix).

4) Web Application: We created a simple web application (React and Flask) in order to better see our model's output for a given input. See github for more details.

V. Discussion

A. Successes and Findings

1) Results from Human Evaluation: As a group we were fairly pleased with the model results, as the output chords seem to be generally related to the melodic content and the chords are generated at times that make some sort of rhythmic sense. However, we found that the expectations from our volunteer were much higher. The volunteer was generally dissatisfied with the generated chord accompaniment and reported each output to be worse than the original accompaniment for the input melody. The volunteer reported an overall average satisfaction of 3.5/10. The reported satisfaction with harmonic choices (which chords are chosen) was 2/10 and the reported satisfaction for rhythmic choices (when the chords change) was 2.5/10.

The volunteer reported more detailed feedback for each output as well. For example, the volunteer found that the generated accompaniment for first flute version of Giant Steps (Giant Steps flute.wav) to be much better than the second version of Giant Steps with the same instrument (Giant Steps flute 2.wav). This suggests that differences in performance can affect accompaniment output.

The volunteer also noted that in some outputs there is a lot of "seemingly unneccesary chord movement for what is a relatively simple tune in reality." There could be many reasons for this, some of which we will discuss here: it could possibly suggest that additional post-processing parameters around chord changes may need to be tweaked in order to account for user preference of chord change frequency. However, it could also suggest that chord change frequency should be better captured by pre-processing parameters. It is also possible that, since the data we trained the models on are mostly pop music but the volunteer input was all jazz music, there may be trending differences in chord change frequency between pop and jazz music. Our chosen methodology does not allow us to argue for the likelihood any of these possibilities (see Limitations), however, these are intriguing

possibilities that could be taken on in the future with properly evaluated machine learning research practices.

The volunteer also reported an average of 3.25 correct chords counted for each song. This is a relatively small percentage of the number of chords produced by the model in each output, which tells us that our volunteer found most of the chords to be "incorrect". This is perhaps due to the fact that the volunteer has ample experience improvising and composing in jazz genres, which could lead to high standards and expectations of specific kinds of accompaniment elements. The volunteer is also extremely familiar with the original songs, which may lead to strong preferences for the original chord progression. More testing would need to be done on newly composed melodies in order to better understand whether this is a general dissatisfaction with our model's output, or rather a preference for the original chord progression.

2) A Tendency Towards E Minor: Additionally, we noticed a tendency for the model to predict E-Minor more frequently than other chords, regardless of the key of the provided melody. This may be due to the distribution of keys represented in the training data. The presence of frequent C-Major or G-Major chords would also partially reinforce a model which skews to E-Minor, due to the shared notes.

B. Limitations

1) Issues with Evaluation Methods: We acknowledge that it is not proper research practice to train our Chordinator model with ground truth labels from our chord detection model without proper evaluation. One of the major reasons we did not use the already available options for chord detection such as MadMom is because they only use chord templates for major and minor. We believe that the inclusion of diminished and augmented chord templates is critical for the success of our model, and so in the spirit of using this final project as an opportunity to experiment, we decided to take the risk of using unevaluated output from our chord detection model as ground truth for our Chordinator model. As a result, this limits our ability to argue exactly why our model produces the given results.

2) Pros/Cons of Using Two Different Model Architectures in One Pipeline: It would have been possible to accomplish training both of these models using the same architecture. Again, in the spirit of experimentation, we decided to go with two different models so that we could get experience working with two different types of architectures. If we were to attempt completing a future research endeavour with this topic, we would opt for similar architectures in both models in order to reduce the number of unknown variables introduced by our chosen methodology.

VI. Appendix

A. Evaluation Song List

Input Files Folder
Output Files Folder

- Giant Steps (jazz)
  - flute (1a)
  - flute 2 (2a)
  - vox (3a)
- 500 Miles
  - alto saxophone (4a)
  - soprano saxophone (5a)
  - flute (6a)
- Solar
  - alto saxophone (7a)
  - soprano saxophone (8a)

B. Evaluation Form Results

Evaluation Form Template (pdf)
Evaluation Form Responses (pdf)
Evaluation Responses Cleaned (pdf of table)

References

[1] Jonathan Adam and Adrian Latupeirissa. Exploring generative models for creating chord progressions. 2017.

[2] Yi Ren, Jinzheng He, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. Popmag: Pop music accompaniment generation. In *Proceedings of the 28th ACM international conference on multimedia*, pages 1198–1206, 2020.

[3] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[4] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. Musdb18-a corpus for music separation. 2017.

[5] Laurent Oudre, Yves Grenier, and Cédric Févotte. Template-based chord recognition: Influence of the chord types. In *ISMIR*, pages 153–158, 2009.