Chordgi

A Musical Recognition Software

James Clark Erica Jurado

December 9, 2017

1 Introduction

Even for the most talented musicians, recognizing chords is both a challenging and time intensive task. This task of classifier for a beginning musician is even more daunting. Despite chord classification being a challenging task, it is one that is often essential to learning music quickly for instruments such as the guitar where chords are key. We built Chordgi, a chord recognition web application, to assist musicians of all skill level to quickly and accurately classify chords. Chordgi uses a trained classifier to quickly and accurately classify musical chords that the user uploads. The user goes to ericajurado.com, clicks on the corgi, selects the .wav file they want to classify, and then they get the musical chord or chords returned to them. For advanced musicians this tool is a great way to save time or check their work. For novices it provides a user friendly way to learn new chords or learn what chords theyd need to play to reproduce a favorite song.

Learning to play music is a challenging task and the initial curve can be offputting to new musicians. Chordgi can help ease new musicians into learning new chords and provide instant feedback to see if they are playing the correct chord. For more advanced players, Chordgi can be a great way to check ones understanding and save time. Most importantly unlike other tools which seek to tackle similar problems, Chordgi is easily accessible on the web, user friendly, and is free so there are no financial obstacles to use.

We are both musicians and have previously worked to train our ears, learn new chords, as well as try to learn what chords are in pieces we like. Although there is a lot of information out there to learn more about music, much is not easy for all levels of musicians to understand or is too expensive to be accessible. We wanted to build Chorgi so this could solve this problem in our own lives. Creating a easy to use application to help identify chords both benefits ourselves

as well as other passionate musicians looking to train their ear, identify chords, or get feedback on their own playing.

Additionally, the technical challenges it presented were ones we were both passionate about tackling and learning more about. Both of us are interested in machine learning and felt that this would be a good opportunity to further our knowledge and get experience applying machine learning techniques in meaningful ways. Furthermore the domains of signal processing and audio technology in general were not ones which we had previously explored, and we wanted to take this opportunity to learn more about it. Finally, we both wanted to get more experience in web development, both front and back end alike. Creating a full stack web application was the perfect opportunity to push ourselves and learn how to make a website and setup our own domain.

2 Proposed Method

To implement an intelligent system for chord recognition we propose applying a novel approach using an Artificial Neural Network (ANN) trained by the backpropagation algorithm. The neural network that will be used for this system will be the Multilayer Perceptron which consist of 3 layers, an input, hidden and output layer. The input for the ANN will consist of 12 features all composed from our Harmonic Pitch Class profile chroma vectors.

2.1 Multilayer Perceptron

A multilayer perceptron is a feedforward neural network that consists of an input layer, an output layer, and one or more hidden layers. What distinguishes the multilayer perceptron from other types of neural networks is that it can be used for a multiple class learning problem.

A multilayer perceptron can be defined as a composition of two maps ϕ and

ψ where the goal is to minimize the error of the produced class labeled variable and its target value.

$$\phi : X \to H$$

$$\psi : H \to T$$

In this given study the network consists of only a single hidden layer which takes the input x $\in R^d$ and maps to h $\in R^p$ with a non-linearity activation function θ.

$$h = \theta(wx + b)$$

then h is mapped to a single varibale which we will denote as $\hat{t}$

$$\hat{t} = (w'h + b')$$

The error produced by the target value and the predicted value can be defined in terms of many ways, for this study we used the the squared error. Multilayer perceptrons are trained to minimize the error defined as the following:

$$L(t, \hat{t}) = ||t - \hat{t}||^2 = ||t - (w'(wx + b) + b')||^2$$

To minimize the loss of the predicted and target value multilayer perceptrons need to be trained through weights adjusting. There are different propagation schemes but for the purposes of this experiment we used gradient descent.

Stochastic online gradient descent parameter updates are performed after each instance has gone through the system

$$w(t) = w(t-1) - \eta \frac{\partial L}{\partial w}(x, w(t-1))$$

Where w consists of both the weights and the biases of the system, n is the

learning rate L is the loss function of a particular instance.

2.2 Harmonic Pitch Class Profile

To be able to turn a sound into numerical values for the computer to understand one must use a method that analyzes the frequencies of a given sound wave. We propose to use a Harmonic Pitch Class Profile (HPCP) which is allows us to search the frequency consistency of an audio clip. The HPCP is built off of ideas originally proposed by (Fukashima 1999) s Pitch Class Profile (PCP).

The way that the HPCP works is that we first run a fast fourier transform (FFT) on our audio signal to convert from time domain to the frequency domain. After this has been accomplished, the next step is to analyze the peaks, if one was to analyze a piece of music you would expect there to be different notes played, which will lead to several different peaks. The HPCP provides a way to find these peaks and analyze them in a FFT plot.

The HPCP algorithm returns a vector of size 12, which represents the 12 notes possible in an octave. The vector is then normalized to represent the likelihood that a particular note is part of the piece of music. These values are computed by

$$HPCP(n) = \sum_{i}^{nPeaks} w(n, f_i)a_i^n$$

Where n $\in$ 1 $\leq$ n $\leq$12, a is the linear magnitude of the ith peak, andf_i is the frequency value of the ith peak, and w is the weight of the frequency.

The weighting function of the above formula is computed in a three step process. First we need to compute the value of f_n which represents the frequency of the note

$$F_n = f_{ref} 2^{\frac{n}{size}}$$

Wheref_{ref} is set to 440Hz. The next step is to minimize the magnitude of the distance d, where the role of m is the drive d to zero so that the difference in a vector is gone.

$$d = 12log_2(\frac{f_i}{f_n}) + 12m$$

The last part of this process is to then compared the distance value obtained by d, by the piece wise function

$$\begin{cases} cos^2(\frac{\pi}{2})(\frac{d}{.5l}) & |d| \leq .5 \\ 0 & d > .5 \end{cases}$$

Where l is the length of the weighting window. In this equation if d is small then we believe there is a correlation between the particular note we are looking at and the peak we had discovered. We measure this correlation through the cos^2 function above, otherwise we set the correlation to 0.

2.3 Preprocessing and Testing Phase

The classification step involves a preprocessing step and a last a identification step. The preprocess state involves frame blocking, windowing, FFT, and last the HPCP algorithm. The classification step involves putting the chroma vectors from the HPCP as input into our ANN for its prediction. The flowchat given in the figure below outlines the processes involved in preprocessing and classifying our data.

In our preprocessing step we used applied a frame blocking process with a size of 4096 which is around .93 milliseconds and applied an overlap of 50% to

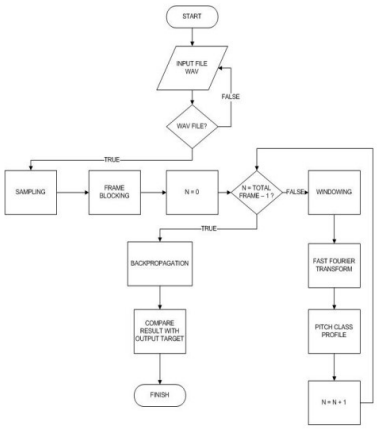

Figure 1: Flow Chart of Backend

each frame. The next step is spectral analysis which involves multiplying each frame by a windowing function and in our case we used a Hamming window. The size of the hamming window is as wide as the frame width. After windowing function is applied we compute a FFT and find the peak or largest value of that frame. Once the peak has been found from our analysis the proposed HPCP algorithm can be applied. Once the vector is applied we normalize to get all of the values between 0, 1 which is a prerequisite for our neural network.

Once the feature vectors have been generated we can train and test our neural network on this input. The parameters of our neural network during the training phase consisted of a hidden layer size of 13 neurons, a learning rate of .1, and a total of 5 epochs on our training data. For the testing phase we tested a total 30 chords at a rate of 95% accuracy. We also tested our network on different types of instruments including, piano, violin, and accordion. The piano rate tested at a 14% accuracy rate, violin at 41% accuracy, and accordion at 34% accuracy.

3 Stack

In addition to developing a classifier using a neural net, we needed to make this classifier accessible. Downloading a repository off of github, figuring out how to compile the code, and figuring out how to get it to run and classify the file you want is not user friendly. To make our classifier accessible to musicians, we decided to make the front end be a website. Initially we had considered making it a phone application, but decided that we wanted to create something that was cross-platform and wanted to avoid making users download anything.

3.1 Backend

Our next step was to design our back-end. The back-end or the server-side of a site is what manages how the site works. This is the portion that the user cannot see like the database and server. When deciding on the structure of the backend we focused on security, structure, and content-management and made sure that what we picked would enable us to create a dynamic site. There were multiple frameworks that we looked at.

The first is Angular. Angulars architecture is a Client-side Model View Controller (MVC). The model and view are linked to allow two-way data binding. Additionally Angular is very modular and encourages an organized back end structure. However, Angular has a steep learning curve and although its best at two-way data binding, for our project we had no need for this feature.

Another framework we looked at is React. React is a high performance client and server side rendering framework. Unlike Angular is only supports one-way flow for data binding. It is also built around making reusable chunks of code to build a website that is easy to maintain. However, React is more of a library than a framework and our project did not require that we use pure JavaScript to create templates.

The final framework we considered and ultimately went with is Node. In short it is JavaScript that can run on the computer using the V8 JavaScript runtime engine. This simplifies development and because the architecture is modular, it is easy to maintain. Additionally, since everything can be written in Javascript, the same Javascript can be used for both the front end and back end development. Additionally, Node allows us to monitor network traffic and reply to HTTP requests. Overall we felt that Node would be the best choice because it is a powerful platform for creating the client-side application we wanted, would be easy to maintain, and gives good service for our application which would ideally have many open connections for a long time.

3.2 Hosting

Once we decided that we we wanted to use Node, we then worked to decide what kind of hosting we wanted to use for our website. Dedicated servers are good for high traffic or for specific setups. Shared hosting is good to save money or if your site does not have high traffic. The final kind of hosting is Virtual Private Server (VPS), which is a virtualized server. This mimics a dedicated server within a shared hosting environment. One of the benefits of VPS is that it is private since you do not share your operating system with others and there are no other websites that have access to your files. Additionally, it enables you to choose your own operating system and have all server instances be your own. You have full control over the system, can make any changes you want, and you can access all of your dedicated RAM at any time.

The features we prioritized in picking a VPS web hosting service were low cost, Linux-based operating system, ability to scale, minimal downtime, good documentation, and solid customer support. Ultimately we went with DigitalOcean due to the very low price, complete freedom to configure your server,

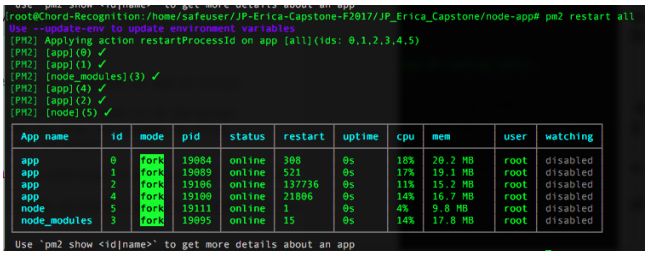

Figure 2:

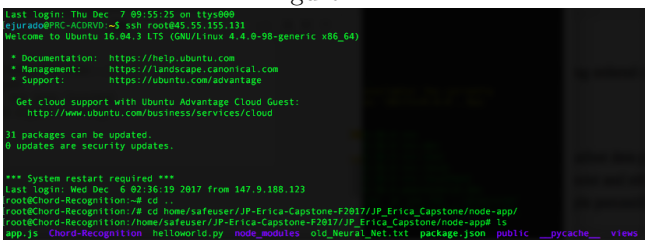

Figure 3:

offer Ubuntu OS, have minimal downtime, and have good articles on how to get your server setup. To purchase our domain, ericajurado.com, we used Google Domains since it is cost effective and easy to manage.

3.3 Full Stack Architecture

On the Ubuntu server (seen in Figures 2 and 3) we set up our Node backend. Node alone, however, is not capable of producing or using a classifier. Using the Node child process module, we were able to execute the python script asynchronously to target the .wav audio file, classify it, and pass the outcome back to the Node backend. Then this information is passed to the client-side to be displayed to the user.

3.4 Front-End Design

One of the biggest goals for Chordgi was for it to be accessible and easy to understand. In terms of accessibility, it is important that the solution be cross

platform so the website can be used on mobile, tablet, or desktop. Initially we thought that our users would be primarily mobile-based, but ultimately decided that musicians, especially novice musicians, would be more likely to be sitting down at home with a laptop or tablet nearby since they would be reading music online or looking up videos. We therefore decided to prioritize desktop and tablet development.

However, our design is one that works well across all platforms since it is clean, simple, and the affordances of all the user-interface (UI) are clear. When the user first opens the page, there is very little on the page. The logo, corgi image, and bottom bar are all visible and the calming, yet contrasting color scheme emphasizes these items. On mouse hover-over, the corgi animates and this responsive element signals to the user that it is something they can click on and interact with. Doing so enables the user to upload their file. Formidable enables us to upload this file to our Node backend. Alternatively the user can click on the Chordgi logo and upload a file that way.

Once the user uploads a file, Chordgi gives the user instantaneous feedback and lets the user know that the file is uploading. This is essential so the user knows that something is happening. If the file is too large, the user is notified that they need to upload a smaller file. Once the .wav file is uploaded, classified, and the client-side receives the classified chords, and displays it to the user. Additionally, the song is played automatically and the sound waves are visualized.

The Web Audio API is used to control audio and add effects through audio nodes. The audio streams within the file and mixed into a single, complicated wave. The front-end draws these waves using bars to represent different indexes of the array making up this wave and updates the canvas constantly to visualize the wave. Visualizing this audio is not only fun for the user, but also adds

another element of interactivity, and allows the listener to follow along and listen to the chords so they can best learn.

4 Obstacles

4.1 Machine Learning

For machine learning techniques to work well one needs to provided an adequate amount of data to make statistical inferences based upon what is given. We faced major challenges due to the necessasity of data and had to develop our own. We were able to develop a small data base of guitar chords based upon the single guitar that we had. If we were able to have multiple guitars and larger data base our network would be capable of handling classification for a variety of different tonal signature. For example an electric guitar has a different type of fingerprint in the audio world than does an acoustic guitar, also every electric guitar is somewhat different from one another.

4.2 Stack Development

The stack for Chordgi also posed numerous challenges. Deciding what the stack should be and going about implementing the back-end was one of the most challenging aspects of the project. Once we decided on the criteria for our backend and VPS, we evaluated numerous options. Implementing these were challenging since it was our first time. In particular, hooking up the domain we purchased, ericajurado.com, to our server posed a challenge. Even when this is setup properly it can take several hours to have these changes be live so it was not something we could instantaneously check. Additionally, our VPS server was headless so changes we needed to make were all using nano and vi without any sort of syntax checking so changes were time consuming.

4.3 Front End

Front-end design also posed a challenge. We had to figure out how to make the UI across platforms seamless and make sure that the affordances were clear. Additionally, the front-end needed to be engaging and fun to use but without detracting from its ability to classify chords for the user. We ultimately decided on simplifying the UI to corgi upload button and building a player that visualized the song. Implementing the Web Audio API to do this was extremely challenging because changing the audio nodes were difficult to do and the documentation was not robust. However, the extra work to implement this paid off since the result was a nicer front-end that was both engaging and informative.

5 Project Evaluation

5.1 Performance /Efficiency

On the back end during the preprocessing and classification phase there are several different algorithms involved to get us to each desired module. During the preprocessing phase the FFT runs at a time complexity of O(nLog(n)), this is where most of the heavy lifting is done. For the neural network this portion is actually rather quick and runs at a time complexity of O(n), making for the total run time of our algorithm O(nLogn) which is acceptable. Since each vector is constant of size 12 the space complexity for our data is at O(n).

Our server on DigitalOcean is the smallest option we could purchase due to budgetary constraints. It is more than adequate for our anticipated traffic in the near future. Although it does not classify as fast as we could, its performance is reasonably fast and our code is clean and efficient and works as fast as it can. In the future if we wanted to improve performance, we could simply buy a nicer

server with more memory and processing power.

5.2 Scalability

Node scales relatively well since it is modular. If we wanted to add additional feature to our site we could with relative ease. Additionally, if we wanted to scale up and serve more users, it would be easy to switch to a more powerful server on DigitalOcean or transfer to a different service.

5.3 Maintainability

Maintainence can always be performed due to the nature of artificial intelligence and specifically machine learning. Over time we can grab more and more data from the input of users and we can continuously update and train our neural network to get higher and higher performance evaluation metrics.

Since Node is modular, it is easy to make changes to Chordgis code. In terms of maintaining the server, PM2 is being used to manage Node. This means that in the event that our backend were to crash, PM2 would automatically restart Node meaning that we do not need to actively maintain and check our server to make sure Chordgi is running properly.

5.4 Reusability

For the backend everything is always reusable. There's no reason as to why we could never run another example through our neural network for classification. Our preprocessing step will always work as long as the file given is a wav file otherwise there are 0 limitations on that algorithm.

The overall stack for Chordgi is very reusable. Not only is Node modular, but each feature we have was also written in a modular and organized manner such that if we wanted to reuse any portion of our stack we could easily do so

and apply it to a different stack.

5.5 Usefulness

Overall Chordgi accomplished what we wanted to achieve. It is an accessible, user-friendly, and fun tool to classify music and learn more about music for musicians of all levels. Whether you're an accomplished musician or a beginner trying to learn the ropes every musician can find a use for Chordgi!

5.6 Limitations

There are many limitations posed on our project as of right now due to the time constraints and nature of a semester long project. Currently our application cannot distinguish the difference between minor and major chords. It also cannot tell if a chord is domant, a 7th, 11th, ect. but we have shown that this is a viable option for chord recognition. If we were to have more time and more data I believe that it wouldn't be an issue in trying to solve some of these problems.

In addition to the limitations of our classifier, our front-end also has room for improvement. Due to time constraints as well as our server, we do not yet allow users to upload files other than a .wav file.

6 Project Demo

The figure shows our program currently up and running on the internet. Anybody can feel free to try it at ericajurado.com.

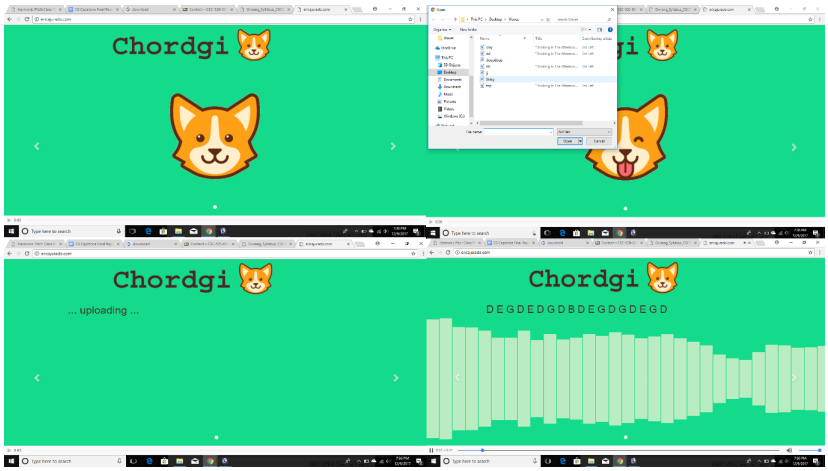

Figure 4: Steps of running

6.1 Future Goals

Although Chordgi was very successful and we made many strides in its development this semester, we hope to further improve upon it. Improvements can be made on all fronts but especially in the classification step. As previously discussed in section 5.6 we can only classify chords as A,B, .. F ect. In the future it could be possible to classify the difference between major and minor maybe by establising more complex machine learning techniques such as bagging, boosting, or one class learning systems.

Additionally, we hope to add additional features. Currently Chordgi only accepts .wav files. In the future we hope to offer support for other forms of audio files and to be able to strip the audio from videos. Other than uploading it directly, we also plan to add support to enable the user to give Chordgi a link to a video, such as a youtube video, and it classify that. We also plan to incorporate a record button on Chordgi so users do not need to pre-record what they want to classify. This would further streamline the users journey.

Finally, should Chordgi take off and we have the financial means to support

it, we would love to upgrade our server to something with more memory and processing power so classification would be even quicker and more users could be supported simultaneously. This would also enable users to upload larger files to classify so they would not be restricted to having only short audio segments classified.

6.2 What we learned

We both learned how to use machine learning techniques in a multi class system, as well as some signal processing techniques discuessed throughout the paper. James had done research in the field of machine learning but only used multiple one class learning systems in a data streaming enviornment. This gave the both of us adequate training and familiarty with the multilayer perceptron. We also established some fundamental principals in the realm of signal processing by introducing such ideas as the fast fourier transform, spectral analysis, and peak detection.

This was our first foray into true full stack development. We both learned how to setup and manage a server. We also had the opportunity to implement a Node application and this helped us better understand server-side development and how it interacts with the client-side through message passing. Additionally, we both learned how domains and DNS works and how to associate a domain with a server. Finally, through the numerous libraries and frameworks used to implement our full stack and make a robust front-end, we also learned how to make better use of APIs, design and implement a full stack solution, and design an attractive front-end.

7 What we achieved

In the course of a single semester we implemented a machine learning classifier that could identify chords and a full stack implementation to build an accessible website, at ericajurado.com. I would like to say that the both of us recieved education in different areas, Erica was able to get more familiar with machine learning techniques and James was able to familiarize himself with some of the front end design and client server full stack development.

7.1 Erica's Main Roles

- Researched back-end frameworks
- Implemented Node framework
- Designed and implemented back-end architecture
- Implemented child process Node module to execute python scripts
- Designed front-end user interface
- Implemented front-end HTML and CSS
- Designed and implemented responsive UI
- Enabled front-end to handle file upload
- Designed and implemented message passing between server and client side
- Used Web Audio API to visualize sound file
- Setup DigitalOcean server
- Setup ericajurado.com domain
- Hooked up domain and server

- Tested and implemented Chordgi on the server
- Setup PM2 to keep Chordgi online
- Maintains server

7.2 James' Main Roles

- Build and design Neural Network
- Create training and testing data
- Research signal processing
- Design HPCP algorithm
- Python development

7.3 Acknowledgements

Our faculty advisor, Kristof Aldenderfer, provided us with invaluable knowledge, constant support, motivational talks, and ensured that we kept up with this project. Additionally he was our inspiration for naming our project Chordgi. His aid helped make Chordgi the fun, useful, and adorable application it is today.

References

[1] S. Baron. Node.js v9.2.1 documentation.

[2] S. Cooke. Node.js foundation.

[3] J. Deng and Y.-K. Kwok. Automatic chord estimation on seventhsbass chord vocabulary using deep neural network. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[4] T. Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *ICMC*. Michigan Publishing, 1999.

[5] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, 2006.

[6] J. Harris. Products on digitalocean — cloud computing for developers.

[7] E. Kerr. How to use pm2.

[8] K. Lee and M. Slaney. Automatic chord recognition from audio using a supervised hmm trained with audio-from-symbolic data. *Proceedings of the 1st ACM workshop on Audio and music computing multimedia - AMCMM 06*, 2006.

[9] B. Markle. What is vps hosting?

[10] J. Oliver. Tutorial: Intro to react.

[11] J. Osmalskyj, J. J. Embrechts, M. Droogenbroeck, and S. Pirard. Neural networks for musical chords recognition. 01 2012.

[12] S. Thompson. Angular tools for development.

[13] S. Walker. Chapter 20 node.js for developers.