A Deep Neural Network-Based Music Key Detection System

Geo Lee

Inquiry Hub Secondary High School
BC, Canada
gogeolee@gmail.com

***Abstract*—This paper presents a deep neural network (DNN)-based music key detection (MKD) system. In recent years, DNNs have demonstrated major success across a wide range of fields as a consequence of the abundance of accessible data for training artificial intelligence (AI). MKD systems are required to detect a particular music key given any piece of music with a dominant major or minor key. The proposed MKD provides further information such as the certainty of the proposed key. The system is composed of four main modules: online music data collection, music key query processing, spectrogram generation, and the neural network model. To reduce the total size of data and more accurately evaluate the correct key of a song, the system converts each song into 40 spectrums to create a popular vote. Experimental results have shown that the system can achieve improved accuracy in the future with refinements not only to the components of the neural network, but to the music theory behind it as well.**

***Keywords*—Music key detection, deep learning, online music data collection, music key query processing, spectrogram generation, neural network model.**

I. Introduction

The incentive for this inquiry was to help musicians without any background knowledge on music theory to quickly determine the key of a sample. Music key detection is not a new concept, but with modern advances in machine learning its development has increased exponentially.

Deep learning is notably proficient with classification problems like recognizing human faces or handwritten digits, so its prospects in recognizing music keys are also promising. It is obvious that learning music theory is also a solution to the problem, but there are still many situations where using an artificial intelligence is convenient.

This paper will go through the steps that created a music key detection deep learning model, and the many factors that contribute to its success.

II. Background

Deep learning is a branch of artificial intelligence which focuses on the ability to learn patterns from data by using neural networks. Simplified, deep learning is a software simulation of brain cells (neurons) with perceptrons. Typically, a single perceptron is created with multiple branches for its inputs, and a single branch for its output. A perceptron can be "trained" to produce a particular output with a given set of inputs by adjusting its weight values.

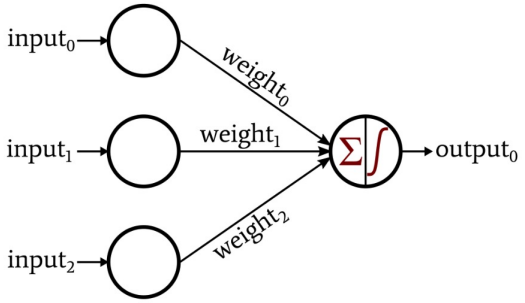


Fig. 1 A simple single perceptron with 3 input branches and 1 output.

As an analogy, each input to the perceptron can be thought as a pipeline of water, and the weight values as valves determining how much water from each pipeline is allowed to go through to the output. The allowed water from each input pipeline is added up, before going through a gate that determines the minimum amount of water required to pass through the output.

It is then possible to form a group of these perceptrons, to make a neural network. There are multiple ways to form neural networks, but the simplest version forms perceptrons into layers, where information flows only in one direction. This is called a "feed forward neural network".

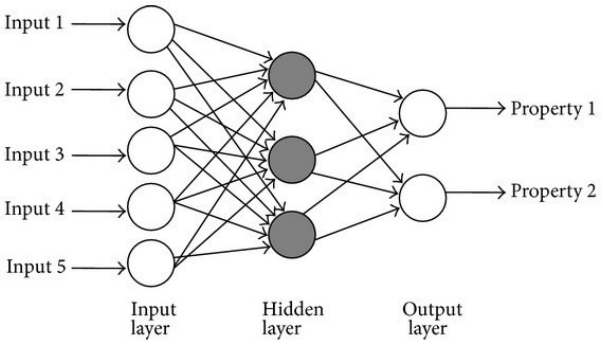


Fig. 2 A simple feed forward neural network, with 5 inputs, 2 outputs, and totaling 3 layers

With multiple layers like Figure 2, the output of a perceptron becomes the input of a perceptron on the next layer (i.e. Input layer > Hidden layer > Output layer),

By adjusting the weights of the neural network (valves of the pipelines), the network can be "trained" to output desired target values given specific inputs. Depending on the size of the data, more layers or more perceptrons are required. Eventually, the end goal is to train the neural network to the point that it can take an unseen input and still give its correct output. For example, if the neural network is trained with 10000 various handwritten digits and their actual intended digit, the neural network should be able to take a newly handwritten digit, and be able to determine what the digit is with high accuracy.

In Figure 3, the image of each digit becomes an input to the neural network, while the actual digit value is the output. For this example, if each file was a 30x30 pixel black and white image, each image can be represented with 900 numbers. These 900 numbers are passed as input values for the 900 input perceptrons of the neural network. Because there are 10 digits to represent, there will be 10 output perceptrons in the output layer. Each perceptron in the output layer represents the probability that the given image file is a certain digit (i.e. First output perceptron represents the probability the image is the digit 1, second output perceptron represents the probability the image is the digit 2, etc)

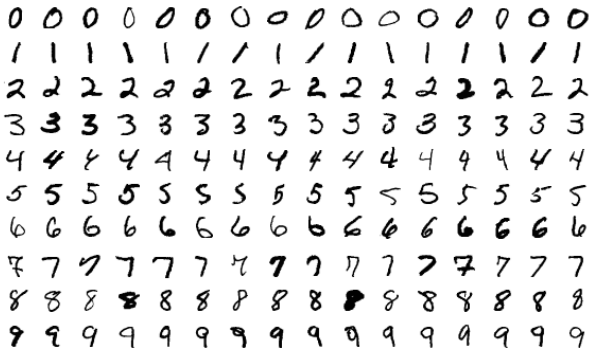


Fig. 3 Sample of handwritten digits

III. DNN-Based MKD System

The system is composed of four main modules: online music data collection, music key query processing, spectrogram generation, and the neural network model. Each module serves an important purpose to the training of the model.

A. Online Music Data Collection

A major selling point of deep learning in recent years has been the large amounts of easily accessible data online. In the case of music, there are large databases of songs through music labels or redistributors. Through making use of Soundcloud and programming python scripts and additional web browser extensions [1] to automate downloading songs from these labels, it is possible to accumulate thousands of songs to use for training. As these music key detection systems are most commonly used by electronic music producers, a specific focus on the subgenres of electronic music was chosen. After compiling a list of song files, songs that did not meet the specific requirements of a modern three minute song were filtered out. This was done by filtering out songs that did not meet a minimum length of 90 seconds or the maximum length of 10 minutes.

B. Music Key Query Processing

Each song is filtered to one of 24 music keys by making a query to the music key database Tunebat. The process of making queries to Tunebat first extracts the song title and artist's names from each downloaded mp3 file. Since each music label titles their songs in different string formats, every label requires a separate script [2, 3, 4, 5, 6]. The module makes an HTTP GET request to Tunebat and parses the HTTP response to find the music key, track title, and artist name. After double checking that the artist and track title match with the request, the program relocates that mp3 file to the subfolder of the particular music key. In this way, the module can filter out a large number of music files into their respective keys from an online database.

C. Spectrogram Generation

To reduce overall size of the input data to the neural network, this module [7] processes each mp3 song into a spectrogram: a series of frequency spectrums over time. To further reduce the amount of unnecessary data, the module only takes spectrums from a small frequency range of 200 Hz to 2000 Hz for each song where the most melodic elements are usually present. Now, each spectrum is represented with 900 32-bit floating numbers, so that every semitone from the frequency range can be depicted. These 40 spectrums with 900 frequency values are much smaller in size than a full mp3 file, and small enough to be passed into a neural net. The module extracts 40 spectrums around the usual chorus/climax section from each song(30-90 seconds is usually where the first chorus is located for most songs). With over 5000 songs and 40 spectrums for each song, the module now outputs 200,000 data entries to use for deep learning.

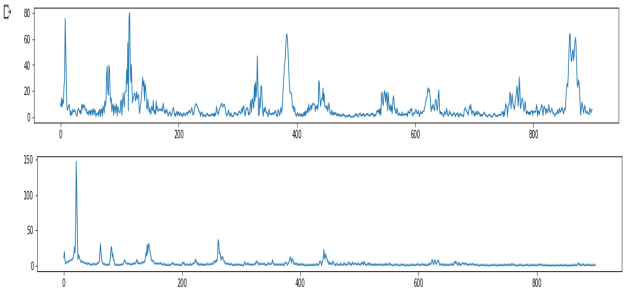


Fig. 4 Sample of spectrums of a song used to train the music key neural net. These are generated and used in place of mp3 files.

D. Neural Network Model

The neural network [8] takes a spectrum of 900 values as its input, and gives the likelihood of each music key in its 24 output perceptrons for 12 major and 12 minor keys. It has a single dense layer of 1024 perceptrons with the ReLu activation function for non-linearity. The model also drops 10% of weights after each epoch of training to avoid overfitting. After 300 epochs with these parameters the neural network managed to build a model with an accuracy

of 76% for each spectrum of a song, and 88% for a song as a whole. Because there are multiple spectrums available for each song, the module passes each spectrum and chooses the music key with the highest occurrence. This has the effect of checking the answer 40 times instead of just once to create a popular vote. As shown in Figure 5, the system can achieve an improved accuracy by fine-tuning multiple parameters.

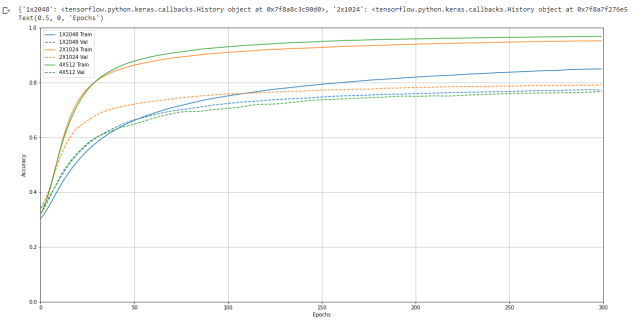


Fig. 5 Graphical representation of three models with different dimensions as they are being trained.

IV. Discussion

As a self taught musician, I have personally found several issues that constantly rose up while producing music. The motivation for this project started as creating something that would help my music production by finding a solution for some of these problems. The project initially started with the idea to create a vocal synthesizer to fulfill my need for an accessible vocalist. Half a year of research into statistics and machine learning through online university courses later, I recognized that vocal synthesis was much too deep for me to completely understand, let alone recreate, with the time left until the project deadline. Choosing instead to narrow down the topic, I delved into deep learning to create this project [**].

There are several issues and potential fixes, as well as future improvements that are worth mentioning.

While this project has managed to find thousands of professional grade songs online through Soundcloud, it has only focused specifically on electronic dance music, meaning it is not attuned for other genres such as classical or jazz. To fix this, a much larger database of songs is required, which means there must be a more automated process to find songs on the internet and sort them by their respective music key. Obtaining a larger database of songs means that the data size of each song must be minimized even further to avoid an irrationally large total data size.

The specific parameters of the spectrogram generation module are variable, and further testing is required to determine the best method of approach. There is a balance of finding the minimal amount of essential data for each song to provide the model proper training. The 200Hz-2000Hz frequency range was chosen to include domains where melodies or chords often lie, while cutting out frequencies that often do not correlate with a particular note, such as drum crashes or low rumbles. Each spectrogram with 40 spectrums covered the 30 - 90 second range, where most drops or choruses of modern electronic music lie. An improvement would be to find and detect drops through their characteristics, instead of fixing specific times for every song. The current ranges were entirely chosen from personal music production intuition.

The current neural network uses parameters that have not been rigorously tested to the extent necessary to determine their effectiveness with relation to other values. These values include the step size for the gradient descent, number of layers for the model, number of perceptrons in each layer for the model, number of epochs for training the model, activation function, and dropout rate are all factors that affect the success of a model.

V. Conclusion

This paper presents a music key detection system based on deep learning with a dataset entirely collected and processed by myself. While it is only a matter of time before technical improvements such as gathering more data can be achieved, there are also fundamental refinements that require a deeper look into the music theory behind the sounds.

Video Presentation

[**] Final project presentation video
https://youtu.be/ISeRo3-F82o

Source Code

[1] A web extension to automatically click download buttons on a Soundcloud page (https://github.com/g30133/soundcloud-clicker)

[2] Tunebat query processor Monstercat: https://drive.google.com/file/d/1hqaH8H1XqQzfxbed5JEKF1QeYljwetzl/view?usp=sharing

[3] Tunebat query processor Musical Freedom: https://drive.google.com/file/d/1Qngq9Qw9CnZ-V0OuFbAz9h4hBOOP8SwF/view?usp=sharing

[4] Tunebat query processor NCS: https://drive.google.com/file/d/1Tzn9NTlD19ZeutruIn7HXAXHaiTOWYMt/view?usp=sharing

[5] Tunebat query processor Spinnin Records: https://drive.google.com/file/d/17easgodfZzA-IQN-CuTwZevSnRb2bdEY/view?usp=sharing

[6] Tunebat query processor MrSuicideSheep: https://drive.google.com/file/d/1Qu7WLHno7xRm7cLul9ghlF0i-uTC5EMN/view?usp=sharing

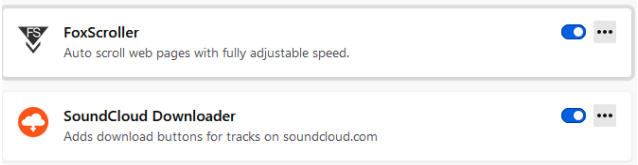
[7] Spectrogram generation https://drive.google.com/file/d/1GDwwiwkA7PQO-BkOdmEjmjNoHJUcFs1n/view?usp=sharing

[8] Final neural network https://colab.research.google.com/drive/19h0WrMg20IMu8drUYX78YMZXNbDipNme

Appendix

Log of Development Milestones

- Found Utau, a free vocal synthesizer.
- Spent a month experimenting with Utau. Made a song, and used Utau to make a vocal topline to accompany it.
- Researched on how Sinsy worked, and found it was very mathematically deep.
- Took a MIT statistics course taught by Professor John Tsitsiklis. https://ocw.mit.edu/resources/res-6-012-introduction-to-probability-spring-2018/index.htm
- Took a MIT mathematics course on machine learning by Professor Gilbert Strang. Was motivated to take another course that focused specifically on deep learning. (https://ocw.mit.edu/courses/mathematics/18-065-matrix-methods-in-data-analysis-signal-processing-and-machine-learning-spring-2018/)
- Researched into Sinsy with what I learned from the 2 MIT courses. I found that the topic was extremely deep. Sinsy was the cumulative work of several top universities over 30 years.
- With the time remaining in the year, I realized that I would not be able to fully comprehend, let alone recreate Sinsy.
- I decided to narrow down the topic to deep learning specifically. I still wanted to work with music in some way, so I decided to make a music key AI.
- Took MIT Introduction to Deep Learning 6.S191(http://introtodeeplearning.com/) and learned to use Tensorflow (deep learning framework).
- Took a deeper look into Tensorflow with its tutorials, specifically examples dealing with audio. (https://www.tensorflow.org/tutorials/audio/simple_audio)
- Started collecting songs for training data. Initially downloaded several hundred songs from the music label NoCopyrightSounds(NCS). (https://soundcloud.com/nocopyrightsounds/tracks)
- Made a python script to automatically take a mp3 file, and find its music key by making a query to Tunebat. I got blocked by Tunebat, because they identified my automated queries as an “attack” on their server.
- Wrote a Firefox web extension using javascript to make the queries to Tunebat not seem automated, and therefore not rejected. Managed to work.
- Trained the neural network using the ~500 songs/music key pairs from NCS and Tunebat.
- The accuracy of the neural network was around 5%, which is around the same as just randomly guessing out of the 24 music keys. I realized that I needed to collect more data.
- Created a Firefox browser extension to automate downloading many songs from Soundcloud. The Firefox extension I made: (https://github.com/g30133/soundcloud-clicker), which requires two additional public extensions in order to work.



- I often ran the code for several hours at a time over many days, downloading songs from big EDM labels from Soundcloud.
  - *NCS*
  - *Monstercat*
  - *Spinnin Records*
  - *MrSuicideSheep*
  - *Trap Nation*
  - *Revealed Recordings*
  - *Tasty*
  - *etc*
- I found there was a way on Python to fake a web request to Tunebat without getting blocked.
- In order to process each mp3 file to extract the filename and artist, I had to create a new Python script for each music label because each label named their files in their own way(i.e. NCS often had [NCS Release] at the end of the filename, Spinnin Records often had [OUT NOW] at the end of the filename, etc). Below are a few of the scripts I created.
- Because I accumulated around 5000 songs, the total size of the mp3 files were too big to be uploaded onto Github. I needed to shrink the size of the total data by shrinking the data size of each song. Instead of uploading mp3 files directly onto Github, I thought of using the spectrums of each song.
- I made a python script to generate a list of spectrums from each mp3 file in order to reduce the data file size for each song. (https://drive.google.com/file/d/1GDwwiwkA7PQO-BkOdmEjmjNoHJUcFs1n/view?usp=sharing)
- Each spectrum is 900x4 bytes. Each song has 40 spectrums. Each song is now only (900x4x40) 144KB. Much less than the average 5 mb of a typical mp3 file(3%)!
- Uploaded spectrum files onto Github, organized by music key. (https://github.com/g30133/data-music-key-spectrogram)
- Wrote a python script to run on Google Colab. It downloads the data from github and makes it into a dataset to be used for my neural network. (https://colab.research.google.com/drive/19h0WrMg20IMu8drUYX78YMZXNbDipNme)
- For each unseen input spectrum, the neural network predicted its music key with 76% accuracy
- Managed to reach ~88% accuracy by using all 40 spectrums from a song to decide its key, instead of just 1. This means that given a song, the probability that the AI tells the correct music key is 88%

```
Popular vote accuracy: 88%
```

- Now, I am working on tuning my neural net (# layers, # perceptrons per layer, training duration) and made a program to test several models at once. Below is a graphical representation of three models with different dimensions as they are being trained. My goal is to eventually reach an accuracy of over ~98%.