Musical Signal Processing through the use of Spectrograms and CNN's

Sean Turland, Gareth Flynn, Jay Pereira, Josh Howells

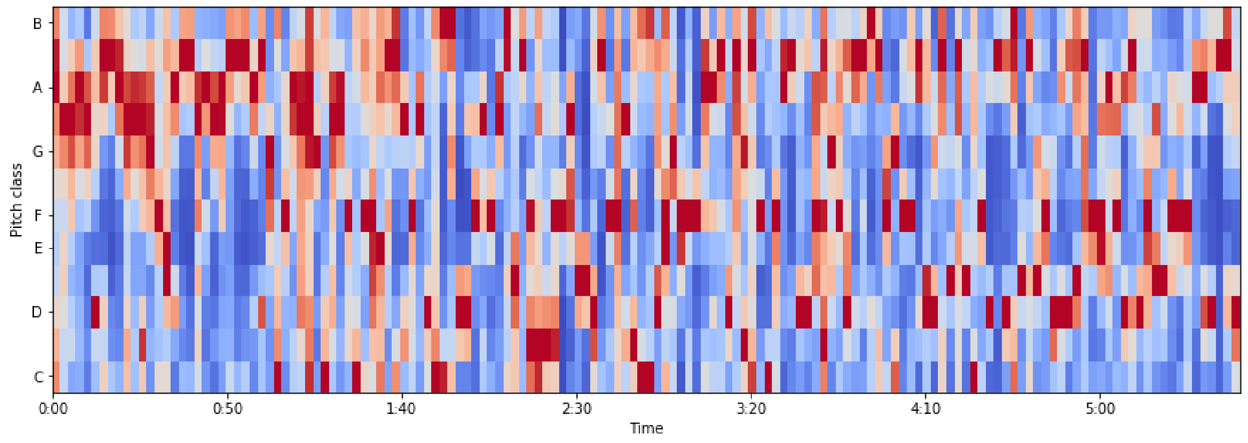

Figure 1: Chromagram of song Freudian Slip - Ulrika Spacek. Relative Weights of frequency bands across time, grouped into the 12 pitch classes (C, C#, D etc.)

Introduction to the problem

Audio files contain a huge amount of extractable data. From a simple vector of amplitude values sampled across time, we can detect everything from low-level features like frequency, pitch and tempo, all the way to its timbre, danceability and even emotion. By training a Neural Network, we can learn which attributes are associated with which genres, with the aim to classify genres of unlabelled songs after analysing their spectral features.

We will convert audio into a spectrogram, which is an image that represents both the time and frequency content of the sound.

Achieving this, it would be interesting to gather our own dataset and derive trends from a set of audio.

For instance, Spotify's Audio Analysis web API can be called in Python using the library Spotipy. From here, we can bypass the computationally-heavy manual audio analysis and instead request their own stored audio data for a particular song.

Literature review

“An evaluation of Convolutional Neural Networks for music classification using spectrograms” by Yandre M.G. Costa, Luiz S. Oliveira and Carlos N. Silla Jr., in Applied Soft Computing 52 (2017).

“Randomly Weighted CNNs for (Music) Audio Classification” by Jordi Pons and Xavier Serra in ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

“Rethinking CNN Models for Audio Classification” by Kamalesh Palanisamy, Dipika Singhania and Angela Yao.

The literature unanimously shows that CNN compares favourably to other classifiers in regard to audio recognition by learning the representation of different musical styles on spectrograms. Thus there is no need for the design of feature extractors as, instead, the model learns itself. Costa et al. (2017) Showed these to work as well as, if not better than handcrafted methods such as using a Support Vector Machine (SMV) with acoustic features like rhythm patterns or histograms.

The importance of the architecture to the classification is demonstrated by Pons et al. (2019). They demonstrate how even non-trained (randomly weighted) CNNs can outperform other methods of classification and even be close to the results of a trained CNN, without the dependence on a large training set.

CNN’s, however, are not explicitly built for use with 2-D spectrograms, so some modification to the architecture is required for optimal performance. There are several methods of achieving this. Palanisamy et al. (2020) were able to demonstrate state-of-the-art performance using simple mel-spectrograms as the sole feature and evaluating the average outputs of an ensemble of independent models to improve accuracy. This reduces time and space complexities of developing a multi-network model, as the CNN model can simply learn the boundaries of the energy distributions in the spectrograms to classify the audio.

Method

1. Gather our own dataset. Python library Spotify enables easy 30-second audio requests from Spotify's API on a per-track basis.
2. Alternatively, the ‘genre dataset’ folder in our GitHub contains a dataset of song metadata already gathered from Spotify. There are over 80,000 entries spanning across every genre, with their pre-calculated audio analysis features (danceability, energy, loudness, pitch etc.) It also includes each song’s Spotify URL, so we can easily request 30-sec audio previews to complete our own audio analysis.
3. Convert audio to a lower sample rate. The vast majority of spectral features will be retained at a lower sample rate, and this has a huge effect on computation time by reducing the number of frames to analyse. Most of the literature uses a sample rate of 22.05kHz as a good trade-off between data and computation.
4. Create spectrograms to represent the audio in visual form. This is achieved by applying a Short-Time-Fourier-Transform (STFT), which takes a Fourier transform at a small number of frames, calculating the amplitudes of each frequency at

that point. A heat map is then created across the frames. See fig. 2 for our initial attempts at converting audio into a spectrogram.

5. Analyse audio files to extract quantitative spectral features. Bandwidth, Zero-Crossing Rate and Chroma can be used by a basic Artificial Neural Network to learn which parameters are typically associated with each genre. (see Initial Experimentation and Analysis for more details)
6. Alternatively, they can be obtained through Spotify's 'Get Audio Analysis Features' API request.
7. Preprocessing to ensure files are in the correct form and split data into training and test sets.
8. Train a Neural Network on the training data. We are going to compare two methods:
    a. Train a simple Artificial Neural Network on a dataset of spectral features, labelled with their genres
    b. Use a Convolutional Neural Network, which takes labelled images (in our case, spectrograms), and classifies the genre based on the pixels of the image. (See Future Work section)
9. Potentially, we may experiment with combining both of these methods alongside Spotify's audio analysis features to create a general model for classifying genre of any song. Although research suggests that CNN's are the most successful approach.
10. Evaluate the model on the test data, which adjusts the weights of the activation functions in the models to improve accuracy using Stochastic Gradient Descent in the CNN
11. Test Neural Network on new, unseen audio from the test dataset.
12. In the end, the aim is to have an accurate Neural Net Model which takes a 30-second song section as input, and classifies it into a genre grouping.

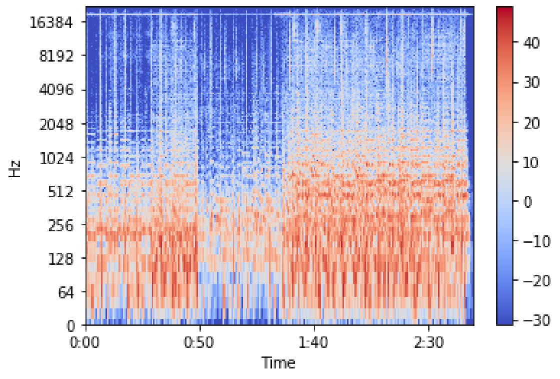

Figure 2: Spectrogram of song Freudian Slip - Ulrika Spacek. The 44.1kHz Audio has been passed through an STFT, amplitude conversion to dB, and plotted on a logarithmic frequency scale

Discussion

Due to the nature and subject of this project, a lot of the prerequisite knowledge for this topic hasn't been featured in the unit. So over the past few months, we've been focused on understanding audio files and how they can be processed, and learning how to best apply neural networks to our data.

Language

Keras is an API built upon TensorFlow that enables fast experimentation with deep neural networks. Alongside PyTorch, it is the industry-standard for deep learning research, enabling immediate adjustment to the models. Using the Sequential class, we can specify each layer within the network to experiment with settings that are best suited to a spectrogram model.

Will also use Librosa, a python library with a focus on musical applications, to convert our audio data into a spectrogram dataset.

Version Control

Alongside using Github to share code and keep a central store of datasets, we've looked into incorporating Spotify's newly released Klio platform as a way to increase our workflow when dealing with masses of audio data. Based on Python and Apache Beam, it's a newly-released open-source framework, where jobs are sent to data pipelines contained inside a Docker image, which would contain the entire job. The idea is to build the model and all its dependencies inside this image, which could easily be shared and reproduced across all of our devices.

Implementing and Extending an Algorithm

To implement a classification model, we have chosen to use a Convolutional Neural Network, trained on images. As mentioned in Palanisamy et al. (2020), this has been shown to achieve equivalent or improved results when compared to a raw audio model.

In short, a convolutional neural network is a spatially invariant deep learning network, which uses convolution in place of standard matrix multiplication. Each layer features a collection of perceptron algorithms with an activation function that, when activated, maps the weighted inputs to the outputs of the neuron. In a CNN, each layer learns a different "texture", with each texture increasing in complexity as the layers increase. These are spatially invariant features, so are searched for across the entire image. This is essential to our project, as features of the spectrogram data (beat, rhythm etc.) will be present at different vectors on the image.

They are typically fully-connected, which means each perceptron neuron in one layer is connected to every neuron in the following layer. The layers will also be pooled, which reduces the dimensionality of the data by combining the outputs of multiple neurons into a single neuron in the next layer.

An advantage of using CNN's is that they require little pre-processing compared to other image training algorithms. The filters that would have to be hand engineered in other ML models are learned by the network,

Originally we considered training a network from scratch, but we have decided to follow the method of Transfer Learning. This involves extending a pre-trained model (trained on a large amount of data), which can already extract useful features relevant to our project.

Baselines - how to measure the success of the model

Requesting a song's metadata from Spotify will give us their own genre classification. Comparing our model's results to Spotify's data will give us an accuracy of the model.

We can also compare our results with other similar research mentioned in the literature review to validate our model's success.

Initial experimentation and analysis

Before we can apply a Machine Learning model to audio, we first had to understand the data format. In our PythonAudio notebook on Github, we have experimented with analysing audio signals. The Python library 'Librosa' is a module that specialises in audio signal processing, with a specific emphasis on musical applications. Sound is represented digitally by a time series vector with an amplitude sampled at each time interval - the length of which is determined by the sample rate. Typically, digital audio has a sample rate of at least 44.1kHz, but for our analysis, we can reduce the sample rate to 22.05kHz without the spectral features being affected to increase efficiency.

These features of the audio can be extracted and used as attributes by a machine learning model to discriminate between data clusters. Some of these features include:

- Spectral Centroid - similar to a weighted mean, it determines which frequency the energy is centred on.
- Spectral Rolloff - a measure of the signal's shape, it represents the frequency at which high frequencies decline to zero.
- Zero-Crossing Rate - a good measure of "smoothness", it represents the rate at which the signal crosses the x-axis
- Mel-Frequency Cepstral Coefficients (MFCC's)
- Chroma Feature - A 12-element feature vector which indicates how much energy of each pitch (C, C#, D,...) is present in the signal. Essentially, it allows the piece to be split up into notes per frequency bin, across the signal. See fig.1 for implementation

To create a Spectrogram, the signal's frequency content is revealed by using a STFT. determining the relevant amplitude of each frequency in a short time window. Spanned across the entire signal results in a spectrogram of the audio (see fig 2.)

To gain an understanding of genre classification and neural networks, we experimented with using these features directly to train a basic Artificial Neural Network on genre classification, closely following "Musical genre classification of audio signals" by G. Tzanetakis and P. Cook (2002). This can be seen in our 'Music Genre Classification' Google Colab notebook on GitHub.

Future work

In the coming months, we intend to gain a better understanding of Convolutional Neural Networks, and extend a pre-trained CNN to apply to a gathered dataset of spectrograms, and evaluate its success.

We also hope to incorporate klio pipelines into our workflow to help manage the masses of data analysis. But so far experiments (and even setup) have been unsuccessful. Getting a full klio job up and running would allow us to send the CNN jobs into a pipeline, which processes in the background through google cloud, but might prove to be outside the scope of this project.

Other project ideas considered

In coming to our final idea, we brainstormed a few other ideas in the process. One alternative project involved creating a stock-trading bot to analyse exchange data and market structure in an attempt to help indicate times to buy/sell a stock based on price action alone. However, we decided that this lacked utility to most people and that the stock market already operates heavily on bot-trading and we would be creating nothing new, with our own flare. In categorizing music, we feel we can create something more unique and interesting, involving a wider variety of outputs besides buying/selling, and the base structure of our project allows for broad further expansion in the future.

We also chose our final idea based on the common interest of music - not only between the members in our group but more importantly in society as a whole. Everyone enjoys music to some degree, and this provides the opportunity to build something meaningful with the wide range of tools and datasets that are available online to analyse audio.

One potential further development to our project would involve using GAN's for use in generating raw audio - involving a generator and sample data being sent into a classifier, which gives its percentage confidence on whether a piece of audio is real or generated. The generator learns over iterations to increase its accuracy, until the classifier is unable to tell the difference between the real and generated data. However, we will mostly be focusing on classifying audio by genre in this course as additions like this are likely beyond the scope of this course.

A discussion on each topic covered in the unit so far & applicability to the project

Relevant Topics:

- Nearest Neighbours - We take our training data which is labelled. A new data point is then compared to the training data to find the k-data points that it is nearest to. The new data point is then classified according to the classification of its closest neighbours.

  In our project, we are using image representations of audio files known as spectrograms. These images have specific features based on qualities such as frequencies, pitch and tempo of the audio file. With audio files of the same genre producing spectrograms with similar features. The k-nearest neighbours' algorithm could be implemented generating a spectrogram of a song we want to classify and then running the kNN algorithm to find the closest k-songs in our training data.

- Convolutional Neural Networks - We can use convolutional networks to classify images. This could be implemented by giving the neural network a spectrogram that we have generated, and the image is then filtered through a series of kernel convolutions. These kernel convolutions will calculate the features of the image, in our case, this will be things like frequencies, tempo, etc, with deeper levels becoming more specific. The CNN would then output a classification/genre for the song.

- Artificial Neural Networks - Simple model used on audio features dataset.

- Stochastic Gradient Descent is used by the activation functions in the CNN to adjust the weights.

Github Link

https://github.com/EMAT31530/ai-group-project-music-classification