Content-Based Music Recommendation using Deep Learning

Ryan Whitell

Regis University

Author Note

This work is dedicated to by brother. He would have thought this stuff was pretty cool.

Abstract

Music streaming services use recommendation systems to improve the customer experience by generating favorable playlists and by fostering the discovery of new music. State of the art recommendation systems use both collaborative filtering and content-based recommendation methods. Collaborative filtering suffers from the cold start problem; it can only make recommendations for music for which it has enough user data, so content-based methods are preferred. Most current content-based recommendation systems use convolutional neural networks on the spectrograms of track audio. The architectures are commonly borrowed directly from the field of computer vision. It is shown in this study that musically-motivated convolutional neural network architectures outperform architectures that are highly-optimized for image-related tasks. A content-based recommendation model is built using musically-motivated deep learning architectures. The model is shown to be able to map an artist onto an artist embedding space where its nearest neighbors by cosine similarity are related artists and make good recommendations. It is also shown that metadata, such as lyrics, artist origin, and year, significantly improve these mappings when combined with raw audio data.

Keywords: deep learning; music recommendation, embeddings, musically-motivated architectures

Contents

List of Tables

List of Figures

Content-Based Music Recommendation using Deep Learning

Introduction

Recommendation systems influence consumers by presenting them with artifacts that consumers are likely to have a positive response to. These systems have broad applications but are primarily employed to increase some form of utility for the consumer which in turn benefits the producer. Some examples of this include consumers being more likely to stay with a certain streaming service that recommends content they like or shopping more frequently at stores that provide useful item comparisons and recommendations. For music services, recommendation can improve a consumer's experience by supporting his or hers discovery of new music, generating favorable radio stations, and curating playlists.

While there may be some concerns over privacy or meddling, recommender systems are generally a positive phenomenon, and their use is becoming ubiquitous online where data are abundant. This research focuses on building a better understanding of the audio signal processing and machine learning concepts that are fundamental to music recommendation systems and ignores any ethical discussions that may be related to the topic (Fule & Roddick, 2004; Van Wel & Royakkers, 2004).

Statement of the Problem

Two main methods are generally used by music recommendation systems: collaborative filtering and content-based recommendation. Collaborative filtering employs user behavior data (liked/disliked tracks, play counts, skip counts, playlists, etc.) to predict what a specific user might like by cross referencing that data with other users. Content-based recommendation attempts to model similarity using only simple descriptors, such as human labeled 'genres' or features derived from Fourier-based analysis on the raw audio signal. The general principals are consistent across different methods, but the specifics or novel approaches to both collaborative filtering and content-based recommendation vary widely and a general best practice has yet to be established.

Most industry approaches to music recommendation combine multiple methods

into so called 'hybrid' methods, which tend to perform better compared to existing methods (Katarya & Verma, 2018). Two popular music streaming services, Spotify and Pandora, serve as good examples of hybrid recommendation systems that tend to favor either collaboration or content-based recommendation. Spotify's hybrid strategy combines collaborative filtering on user behavior with natural language processing (to gain insight into the context of music by scraping the web) and deep learning (to understand the qualities of raw audio signals) to infer similarity (Ciocca, 2017). Pandora keeps comprehensive music metadata in the form of about 400 descriptive 'genes' carefully chosen by domain experts that describe a particular piece of music from which similarity can be measured (Howe, 2009). Spotify and Pandora have both been successful financially, and anecdotal evidence suggests that their users are responding positively to their recommendation services (Ciocca, 2017; Howe, 2009). However, both of these approaches to music recommendation can still be improved.

Music recommendation systems currently face a few challenges. The main challenge for collaborative filtering is the cold start problem, which is when a new user or track is introduced into the ecosystem. There will not be enough data to provide accurate recommendations in these cases. Another more nuanced example of the cold start problem can occur for listeners with unique or niche tastes in music. When there are not enough similar listeners (a threshold of similar users in some niche is not reached) recommendations will typically trend towards popularity. It may be slow or impossible to generate enough data for certain niches to benefit, so these users never really get their 'start.' This phenomenon is associated with the *long tail*- a small percentage of tracks generate the most play counts (Anderson, 2004).

Content-based recommendation comes with its own set of challenges, the main one being the lack of a ground truth. Genres are a good example. Genre labellings are meant to group similar sounding music together, but a study done by Gjerdingen and Perrott (2008) found that assigning a music clip as one of ten genres proved difficult for most people. Individual perspectives on music vary widely and content-based approaches will always suffer from data that lack the full description. Cultural,

emotional, and contextual factors can influence a person's perception of music. If a perfect measure of raw audio content similarity existed, two very similar songs recommended could still be perceived by someone as completely different: for example, suppose one song happened to bring up a positive memory for one person and did the opposite for another person. This would be a bad recommendation for the second person. Content-based similarity measures of something as complex as music is still a difficult problem.

Purpose of the Study

State of the art music recommendation systems use hybrid collaborative filtering and content-based recommendation methods but still struggle to classify and recommend newly-introduced or niche music. Because content-based methods are well suited for solving the cold start problem, they are the the focus of this study. More specifically, this study investigates different content-based approaches to genre recognition and recommendation. Multiple Fourier-based representations of audio are compared. The effects of metadata on recommendation are also explored. Finally, different deep learning models are trained and tested. The results of experimentation are used to propose a content-based recommendation model.

Significance of the Study

Because deep learning for content-based music recommendation is still a relatively new area of study, an investigation of different deep learning methods may further the development of the field. Recent deep learning breakthroughs in the fields of computer vision and natural language processing are translatable to content-based music recommendation. However, a direct translation is not ideal, so musically motivated deep learning architectures are proposed. This study also attempts to gain insight into how song lyrics, album artwork, song release year, and artist origin affect music recommendation.

Literature Review

Recommendation resides in the domains of data science and machine learning, but music recommendation also requires audio signal processing and musicology (for understanding the domain). More broadly, music recommendation can draw from branches of psychology, physiology, philosophy, and anthropology. Understanding the human connection to music is an ongoing research effort that spans many branches of science. An algorithm that truly understands music will require much more research in these areas. Currently, the state of the art recommendation systems use hybrid collaborative and content-based filtering methods, leveraging massive pools of data to make recommendations. Methods that take into account cultural, social, contextual, or emotional aspects of music are beginning to appear in the literature, but these avenues are new and have not been explored fully.

Early Content-Based Methods

Content-based recommendations leverage the simple concept that there are intrinsic similarities between some songs and obvious differences between others. If a user listens to a certain song then songs that are 'close' to that song by some measure make good recommendations, depending on the measure. Music similarity is a continuing research effort that currently relies heavily on the field of music information retrieval (MIR), which is concerned with the extraction of useful information from raw audio signals. One popular sub-task of MIR in the literature is music genre recognition (MGR): classifying an audio signal (with or without metadata) into one or more genres. MGR, MIR and content-based methods face interesting challenges. Datasets for MIR are hard to generate due to copyright. Most datasets used in MGR are from private collections, so results are hard to replicate (Sturm, 2012b). Also, using genre labels as ground truth is flawed because genre itself can be dynamic or vague; the boundaries between genres are increasingly becoming blurred (Patch, 2016). Despite these challenges, progress has been made in extracting useful features from raw audio and metadata to construct similarity measures, clusters, and labels.

Features. The early literature of MIR has been heavily focused on discovering which features derived from audio signals are most emblematic of the music and somewhat focused on which algorithms are best at understanding or applying these features. The features extracted fall (very) roughly into three main categories: low-level, rhythmic, and tonal. Early work done by Tzanetakis and Cook (2002) used features of these three categories for MGR and is frequently cited. Other feature categories exist as well, such as high-level descriptors (e.g. 'danceability') (Laurier, Herrera, Mandel, & Ellis, 2007; Maillet, Eck, Desjardins, & Lamere, 2009).

Audio features. A common technique for analyzing a raw audio signal is to use the Discrete Fourier Transform (DFT) to get a spectral representation, the amplitude and phase of the signal in the frequency domain. The Short-Time Fourier Transform (STFT) is used to get a time-varying spectral representation, a spectrogram. Timbral characteristics of sound can be derived from the spectrum.

The *spectral centroid* is a common feature extracted from the spectrum. It is the location of the center mass of the magnitude of the spectrum and is a characteristic of the spectral shape of a sound. It corresponds roughly to the 'brightness' of the sound. Another measure of spectral shape, the *spectral roll-off frequency*, is the frequency of which some threshold of energy falls below it. It is calculated by splitting the spectrum at some energy interval, usually around 85%, where sounds above it are more likely to be noise and sounds below it are more likely to be important harmonics. The *spectral flux* is a measure of how much the signal is changing and is a characteristic of timbre; it is also used for onset detection. The *zero-crossing rate* is a time domain feature that corresponds to the amount of sign change in a signal, which can indicate percussiveness or noise. These features are used in the literature for MGR with varying results (Benetos & Kotropoulos, 2008; Laurier et al., 2007; T. Li, Ogihara, & Li, 2003; McKinney & Breebaart, 2003; Peng, Li, & Ogihara, 2007; Tzanetakis & Cook, 2002).

A more complex representation of the spectral shape of a sound is the *Mel-Frequency Cepstral Coefficients* (MFCCs). The MFCCs are derived using the mel scale, a close approximation of the human auditory response (Stevens, Volkmann, &

Newman, 1937). MFCCs can represent complex characteristics with low dimensionality; typically, only the first 13 coefficients are used. MFCCs were used mostly for speech recognition until Logan (2000) popularized the idea of using MFCCs for MIR, of which many followed (Aucouturier & Pachet, 2002; Aucouturier, Pachet, & Sandler, 2005; Benetos & Kotropoulos, 2008; Berenzweig, Logan, Ellis, & Whitman, 2004; Bertin-Mahieux, Eck, Maillet, & Lamere, 2008; Flexer, Schnitzer, Gasser, & Widmer, 2008; Karpov & Subramanian, 2002; Laurier et al., 2007; Lee, Shih, Yu, & Su, 2007; T. Li et al., 2003; T. L. Li & Chan, 2011; Maillet et al., 2009; Mandel & Ellis, 2005; McKinney & Breebaart, 2003; Pampalk, 2005; Pampalk, Flexer, & Widmer, 2005; Peng et al., 2007; Shao, Xu, & Kankanhalli, 2004; Tzanetakis & Cook, 2002).

The first MFCC is a measure of energy and is normally excluded and obtained by other means. The energy of an audio frame can be measured in the time domain or in the frequency domain and can be an indication of activity, loudness, or perceived intensity (Benetos & Kotropoulos, 2008; T. Li et al., 2003; Maillet et al., 2009; McKinney & Breebaart, 2003; Tzanetakis & Cook, 2002). Laurier et al. (2007) found loudness to be particularity good at discriminating between some moods for a mood classification task.

Rhythmic characteristics of music like beat and time signature can be estimated through various methods that detect salient periodic patterns (T. Li et al., 2003; Tzanetakis & Cook, 2002). Beat tracking and onset detection are useful tools for detecting segments within a song. Music made specifically to dance to, ballroom music for example, will have a high correlation with tempo. However, rhythmic features are rarely used on their own in the literature for MGR since most popular Western music is in common time and distinguishing genre by beat and tempo alone is nearly impossible. Rhythmic features are mainly used in conjunction with other features.

Tonal descriptors are generally the most abstract or complicated to extract. These features attempt to describe harmony and pitch through various methods (T. Li et al., 2003; McKinney & Breebaart, 2003; Tzanetakis & Cook, 2002). Tonal descriptors can predict with some certainty the key (Laurier et al., 2007), scale, sequence of chords, and

the predominant melody of a song. Lidy, Rauber, Pertusa, and Quereda (2007) used tonal and rhythmic methods to produce a symbolic representation of raw audio in the form of MIDI information, which improved genre classification accuracy over using timbral features alone. Müller, Kurth, and Clausen (2005) showed that chroma-based features work well for identifying different interpretations of the same piece of classical music because they capture melodic and harmonic characteristics while being robust to variations in timbre, tempo, and dynamics.

Other audio features found in the literature include constant-q transforms (CQT) (Schörkhuber & Klapuri, 2010), octave-based spectral contrast (Jiang, Lu, Zhang, Tao, & Cai, 2002; Lee et al., 2007), discrete wavelet transforms (DWT) (T. Li & Ogihara, 2004; T. Li et al., 2003; Peng et al., 2007), linear prediction cepstral coefficients (LPCCs) (Karpov & Subramanian, 2002; Shao et al., 2004), autocorrelation coefficients (Maillet et al., 2009), and dissonance (McKinney & Breebaart, 2003).

Metadata. Generally, two types of metadata can describe any given song: subjective and true. True labels are anything that is not subjective, i.e. artist, album, year, lyrics, etc. Subjective labels include things like tags and genres. Tags can be any subjective labels given to a song, artist, or album, usually as words or short phrases. In most databases, tags are applied by users. This can make tags noisy, especially if the system does not restrict what a tag can be. In these cases, tags are often genres and sub-genres. Genre is the most common source of ground truth for recommendation and MIR tasks, but tags can be used as well (Bertin-Mahieux et al., 2008). Web scraping for information is another way to get more contextual, social, or emotional metadata, such as from critical reviews, social media contexts, or information from wikis (Berenzweig et al., 2004).

Methods. Just as there were many different audio features or representations to consider, the methods by which to apply these features in the pre-deep learning literature varied widely with no best practices emerging. A few examples include Gaussian mixture models, used by Aucouturier and Pachet (2002), Aucouturier et al. (2005), Berenzweig et al. (2004), Jiang et al. (2002), and Pampalk (2005), non-negative

tensor factorization (an extension on non-negative matrix factorization (Holzapfel & Stylianou, 2008)), used by Benetos and Kotropoulos (2008) and Panagakis, Benetos, and Kotropoulos (2008), and hidden Markov models, used by Karpov and Subramanian (2002) and Shao et al. (2004). Even simple designs like using Euclidean distance as a similarity measure showed promise during this time (T. Li & Ogihara, 2004).

The number of features (dimensionality) from audio analysis can depend on the size of the feature window and how far apart these windows are placed (hop size). Window level features are sometimes aggregated by means and variances, or by the running means and variances of previous windows (Tzanetakis & Cook, 2002). For a specific example, Bertin-Mahieux et al. (2008) used features gathered with a window size of $100ms$ every $25ms$ and compressed them into means and standard deviations over every 50 windows. Other statistical descriptors of features can include median, skewness, kurtosis, minimum values, and maximum values (Lidy et al., 2007). Mandel and Ellis (2005) make the assumption that "songs with the same MFCC frames in a different order should be considered identical" and use a 'bag of frames' approach to MFCC features. Bergstra, Casagrande, Erhan, Eck, and Kégl (2006) compressed the window-level features into song-level features by fitting independent Gaussians to each feature. Windows, hop sizes, segmentation, and compression strategies vary in the literature and are influenced by computation and memory constraints.

Most of the audio feature-based efforts limited audio samples to around 30 seconds mainly due to the availability of data and the restriction placed on most machine learning model inputs. However, Aucouturier et al. (2005) went a step further from their earlier work (Aucouturier & Pachet, 2002) and used the entire length of the song, segmenting it into "sections of homogeneous timbre." Their results were not definitively better than those from their earlier research using shorter audio clips; however, in both cases the evaluation methods were qualitative. Mandel and Ellis (2005) did show advantages of using full tracks instead of 30 second clips. Song segmentation techniques are more effective when using full tracks. Shao et al. (2004) segmented songs not by timbre but by rhythmic structure, which they argued "captures

the natural structure of music genres better". Segmentation is not fully explored in the literature and could be a useful research direction.

Ever-increasing computing power allowed many researchers to improve upon past results by using machine learning algorithms like K-Means (Berenzweig et al., 2004; Peng et al., 2007; Tzanetakis & Cook, 2002) and KNN (Tzanetakis & Cook, 2002) for music classification using metadata (artist, genre, year) alongside audio features, or AdaBoost for general music classification (Bergstra et al., 2006). FilterBoost, a similar algorithm to AdaBoost, was used by Bertin-Mahieux et al. (2008) as an 'autotagger' for automatically assigning social tags to tracks using audio features. Support Vector Machines were used for music mood classification (Laurier et al., 2007), genre classification (T. Li et al., 2003; Lidy & Rauber, 2005; Wang, 2016), artist classification (Mandel & Ellis, 2005) and emotion detection (T. Li & Ogihara, 2004). Artificial neural networks were used to recommend music with some success (Oord, Dieleman, & Schrauwen, 2013). Some more unique approaches found include collaborative filtering of radio station playlists for custom playlist generation (Maillet et al., 2009), latent Markov embedding for playlist prediction (Chen, Moore, Turnbull, & Joachims, 2012), and genre classification using song lyrics (Bou-Rabee, Go, & Mohan, 2012; Mayer, Neumayer, & Rauber, 2008).

Results. While it should stand to reason that MGR classification accuracies across the corpus of the early experimental work should be easy to compare, this is not the case. Most of the experimental work used private datasets where replication is not possible. The lack of good public benchmarking datasets is evident by the widespread use of the *GTZAN* dataset (Tzanetakis & Cook, 2002), which is shown to contain replicas, mislabeling, and distortions (Sturm, 2012a). Therefore, accuracy results of MGR systems that use the *GTZAN* dataset, of which there are many, should be viewed with skepticism. However, that does not mean that the discipline had not moved forward. Feature engineering was the focus of early experimentation and many novel music-related features extracted from raw audio were proposed and tested. The early MIR/MGR and music recommendation literature was heavily-focused on this front, and

some clearly superior features, such as MFCCs, emerged as a result.

Modern Approaches

While the early work has been focused on feature engineering and machine learning, more recent approaches are focused on deep learning; this has the advantage of learning which features are most suitable for a task by using a deep, hierarchical structure. In the case of end-to-end learning, the raw waveform of a song can be used as input to a deep model without the need to engineer any features at all (Dieleman & Schrauwen, 2014; Thickstun, Harchaoui, & Kakade, 2016). The number of deep learning related articles represented at conferences of The International Society for Music Information Retrieval (ISMIR) increased from 2 in 2010 to 16 in 2016, a trend that is observed in the fields of computer vision and natural language processing as well (Choi, Fazekas, Cho, & Sandler, 2017).

Deep neural networks (DNNs) *have* been used in the literature (Jeong & Lee, 2016; Sigtia & Dixon, 2014), but convolutional neural networks (CNNs) (Costa, Oliveira, & Silla Jr, 2017; Oord et al., 2013) and recurrent neural networks (RNNs) (Choi, Fazekas, Sandler, & Cho, 2017) are much more common. Most content-based MIR research focuses on audio-based features only, leaving metadata out. Metadata is rarely mentioned and even more rarely the focus. However, metadata focused research does exist in the literature. Oramas, Nieto, Barbieri, and Serra (2017) reported that the addition of album artwork and album reviews to an audio-based model improved the results of a music genre classification task. Additionally, both Bou-Rabee et al. (2012) and Mayer et al. (2008) show that using lyrics can improve the accuracy of music classification tasks.

Theoretical Basis

Music Theory

The basic elements of music are rhythm, melody, harmony, timbre, and texture. The main purpose of Western music theory is to "describe various pieces of music in terms of their similarities and differences in these elements" (Schmidt-Jones, 2013). Music can be grouped into genres based on these elements similarities and differences; therefore, effective music recommendation and MIR/MGR tasks require domain knowledge in music theory. The following concepts should be kept in mind when developing deep learning architectures for music applications.

Rhythm. *Rhythm* can be best described as the flow of music or as the duration and placement of pitch (notes) and silence (rests) in time (Schmidt-Jones, 2013). It is a set of regular or irregular patterns that are usually built off the *beat*. The beat is the underlying steady and regular pulse of a piece of music; it is what one would tap one's foot to while listening. It is also what allows people at a concert to clap in synchronization with each other during a musical performance. The speed of the beat is called the *tempo*, which is measured in beats per minute (bpm). Tempo can help set the "character or mood" (Clendinning & Marvin, 2016) of a piece of music. Music played slowly can "can impart a feeling of extreme somberness," whereas music played quickly can seem "happy and bright" (Pilhofer & Day, 2015).

Rhythm and beat are described using a *time signature*, which specifies how many beats are present in each *measure* (a division of music, also called a *bar*), where a note value constitutes one beat, and "provides a framework of strong and weak beats against which the rhythms are heard" (Clendinning & Marvin, 2016). Music in different time signatures can "feel different, both to listen to and to play" (Pilhofer & Day, 2015). Western genres like rock, jazz, and country are usually written in 'common time,' but, some genres use more complex time signatures like 'math rock.' Time signatures of non-Western music can be highly irregular as well (Pilhofer & Day, 2015). More complex rhythmic analysis includes changing tempos, complex time signatures, syncopation, anacrusis, and hemiola.

Melody. The *melody* is simply a sequence of notes or differences in pitch across time. The melody is usually musically satisfying and memorable; it is the part of a song one would hum (Pilhofer & Day, 2015). A melody may rise and fall slowly with small pitch changes between notes (conjunct), or quickly with large pitch changes (disjunct) (Schmidt-Jones, 2013). A group of notes that fit together and express a melodic 'idea' is called a *melodic phrase* (Schmidt-Jones, 2013). Even shorter than a phrase is a *motif* which can occur frequently with or without variation in a piece of music (Schmidt-Jones, 2013). An artist's use of melody is unique, and some melodic techniques are common across genres. In much popular Western music for example, songs usually contain two main melodies (the chorus and the verse) and will sometimes contain a third (the bridge). Although separating chorus from verse by melody is a simplistic example, audio analysis of melodies can help detect the *form* of a piece of music.

Harmony. A *pitch* is a complex sound wave that has a certain *fundamental frequency* and many overtones. The amplitude of the wave is the *loudness* of the pitch (measured in decibels), or in musical terms, the *dynamic level*. A pitch is an *octave* higher than another if it is twice the frequency; it sounds like a 'higher' version of the same note. A *tone* or a *note* is simply a sound with a particular pitch, duration, and loudness; instruments are designed to produce tones. *Harmony* is the result of more than one pitch sounding at the same time. Each overtone over the fundamental frequency of a pitch is also an example of harmony, as the overtones are part of a *harmonic series* where each overtone frequency is an integer multiple of the fundamental frequency. Harmony is one of the most complex and highly-developed elements of Western music theory (Schmidt-Jones, 2013).

Although harmony is a broad and complex concept, it can be summed up abstractly as which pitches sound 'good to our ears' or how pitches 'go together' mathematically. This manifests itself in Western music in concepts like balancing consonance and dissonance, intervals, scales, major/minor keys, the circle of fifths, and chords and chord progressions. Similar songs may share similar harmonic structure. For example, music in a major key tends to sound more cheerful and exciting, whereas

music in a minor key tends to sound more sad and solemn (Schmidt-Jones, 2013). Specific chords or scales can be more common in certain styles of music as well.

Timbre. Every musical instrument sounds unique (irrespective of pitch, duration, or loudness) because of its *timbre*. A sound wave from an instrument is made up of three fundamental parts: *attack*, *harmonic content*, and *decay*. The attack is the initial sound made from an instrument and it is "the most distinguishing aspect of a note" (Pilhofer & Day, 2015). A sound wave's harmonic content is all of the different frequencies of which it is composed. The human ear does not hear all the separate frequencies; instead, it hears the whole complex wave as the 'color' of the sound (Schmidt-Jones, 2013). The decay is the rate at which the sound dissipates. A plucked guitar string will ring out, while a note played on a flute is sustained. Timbre can influence the 'feel' of a piece of music. For example, guitars with metal strings sound crisp and aggressive, while guitars with nylon stings sound softer and more 'mellow' (Pilhofer & Day, 2015). Entire songs can impart different feelings or emotions by use of "timbral variation" (Clendinning & Marvin, 2016).

Texture. *Texture* refers to the "number and alignment of individual voices or instrumental lines in a composition" (Clendinning & Marvin, 2016). It is how much is 'going on' or how many layers there are in a piece. The overall quality of sound is determined by this layering. Some terms used to describe texture include thick, thin, bass heavy, and rhythmically complex (Schmidt-Jones, 2013). There are three main types of texture. *Monophonic* textures contain only one melodic line, like one hand playing a melody on piano or someone whistling a tune (Schmidt-Jones, 2013). *Homophonic* textures are the most common in Western music. They contain one clearly melodic line with accompaniment or chords (Schmidt-Jones, 2013). *Polyphonic* textures have more than one independent melody occurring at the same time, like in a round (Schmidt-Jones, 2013).

Audio Signal Processing

STFT spectrograms. The STFT spectrogram is a time-frequency representation of an audio signal derived by iteratively applying the DFT on a small

window and then shifting it across the time domain by some hop length. The output is a complex-valued matrix with the amplitude and phase of each frequency at each time frame. In essence, it is a way of extracting the contributions of the frequencies present in a signal. The spectrogram is a more intuitive representation of the raw waveform. Four different audio clips are presented to highlight this fact. While some information may be gained from the phase spectrum, it is ignored throughout this study where only the amplitude spectrum of the STFT is considered.

Figure 1 shows both the amplitude spectrum of a short piano phrase of five notes played staccato and the amplitude spectrum of a cello double-stop where the open D string remains constant below a short seven note phrase on the A string. The fundamental frequencies and harmonics of each piano note are clearly defined and, because it is played staccato, the attacks and segments of each note are clearly visible in the spectrogram. For the cello double-stop, the fundamental frequencies and harmonics of both the D and A strings can be seen clearly as well, even though the notes are being played together.

Figure 2 shows both the amplitude spectrum of a seven second clip of a song labeled 'blues' in the *GTZAN* dataset ("I'm In The Mood" by John Lee Hooker) and the amplitude spectrum of the sound of the ocean waves crashing off Miami Beach. Even a homophonic song like Hooker's is represented well by the STFT because music is structured sound. The structure of music can be better appreciated when compared to a stochastic signal like the ocean sound. The cello, piano, and ocean sound were obtained from *freesound.org* (Font, Roma, & Serra, 2013).

Mel-scaled STFT spectrograms and MFCCs. An STFT spectrum can be mapped onto the mel basis, resulting in a spectrum that scales in a logarithmic way to more closely match the human auditory response. The frequency bin sizes increase as frequencies increase. The idea is that a human can better recognize differences at lower pitch ranges than at higher pitch ranges, an ability that scales logarithmically. For example, a human can more easily tell the difference between pitches at $100Hz$ and $200Hz$ than at $10000Hz$ and $10100Hz$, even though the difference in pitch in each

example is $100Hz$. Figure 3 presents a comparison of the STFT spectrogram and its mel-scaled counterpart. A machine learning model may be able to more easily understand a mel-scaled STFT than an unscaled STFT.

The amplitudes of the spectrum obtained by taking the discrete cosine transform of the mel-scaled STFT are the MFCCs. This 'spectrum-of-a-spectrum' representation describes in low dimensionality the overall shape of the spectral envelope. MFCCs are commonly used in machine learning tasks involving audio. For music applications, MFCCs are related to the timbral and textural quality of music. Given a mel-scaled STFT, a deep learning model should be able to learn this representation on its own if it were deemed useful. The first 12 MFCCs of a short piano phrase are shown in Figure 4.

Constant-Q transforms and chromagrams. Similar to the mel-scaled STFT, a constant-q transform (Brown, 1991) groups frequencies into logarithmically spaced bins. The constant-q transform is a more musically-motivated transformation, however, where frequencies are binned by the frequencies of musical notes. For example, C_1 on a commonly-tuned piano is about $32Hz$ and C_7 is about $2093Hz$. Frequencies can be binned logarithmically the same way that notes are spaced out. The frequency difference of one semitone between lower notes is much less than that of higher notes. The transformation groups frequencies into the same number of bins per octave so that the lower octaves will have higher frequency resolutions. Figure 5 confirms which notes were played in the cello double-stop audio file by plotting the constant-q spectrogram using a 'Note' axis. Essentially, the constant-q transform acts as a 'note identifier.'

As stated before, a pitch is an *octave* higher than another if it is twice the frequency; it sounds like a 'higher' version of the same note. In Western music theory, the pitch classes are: C, $C^{\#}$, D, $D^{\#}$, E, F, $F^{\#}$, G, $G^{\#}$, A, $A^{\#}$, B. Wrapping the constant-q spectrum into these 12 pitch classes results in a chromagram. For example, the frequencies $A_1(55Hz)$ and $A_2(110Hz)$ would fall into the same bin. Similar to how MFCCs are a lower-dimensional representation of the mel-scaled STFT, a chromagram can be thought of as a lower-dimensional representation of a constant-q transform (or an STFT). Instead of capturing timbre and texture like MFCCs, chromagrams can capture

harmony and melody. They are robust to changes in timbre and texture; a melody played on two different instruments would have the same chromagram. Chromagrams do, however, have to make assumptions about the tuning and temperament of a piece of music. The chromagram of a cello-double stop is shown in Figure 6.

Methodology

Two main hypotheses drive the following experiments. The first hypothesis is that CNN architectures exist that are better or more efficient for music-related tasks than for image-related tasks. The second hypothesis is that raw audio data are not representative enough for machine learning models to make acceptable recommendations. Metadata, including but not limited to lyrical data, artist geographical origin, and song release year, are needed to provide emotional, cultural, and historical context when making recommendations. The hypotheses will be tested by training and evaluating various machine learning models on three different datasets. Another goal of this Thesis is to build a content-based recommendation model.

All work was done using the Python[1] (version 3.6.8) programming language. Machine learning models were defined and trained using the Keras[2](version 2.2.4) API with a TensorFlow[3] (version 1.12.0) backend on a NVIDIA GeForce GTX 1080 Ti GPU. The audio signal processing library LibROSA was used for extracting features from raw audio.[4] All code can be found in the GitHub repository.[5]

Datasets

The Free Music Archive dataset. The Free Music Archive (*FMA*) dataset (Defferrard, Benzi, Vandergheynst, & Bresson, 2017) is a collection of 106, 574 Creative Commons-licensed high-quality audio tracks. The dataset is split by the authors into three main datasets (small, medium, large), each split into 80% training, 10% validation, and 10% testing sets filtered for artists to be represented in one set only to avoid the producer effect (Flexer, 2007). The medium subset contains 24, 976 tracks, each labeled with one of 16 different genres, and it is used for a multiclass classification task. The large subset contains 104, 284 tracks, each labeled with any number of 161

[1] https://www.python.org/

[2] https://keras.io/

[3] https://www.tensorflow.org/

[4] https://librosa.github.io/librosa/

[5] https://github.com/RyanWhitell/Deep-Learning-for-Music-Recommendation

different genres, and it is used for a multi-label classification task. Although this dataset contains some metadata for each track, only the 30 second audio clips and genre labellings are used. To foster future comparisons and reproducibility of this work, the suggested subsets and splits are adhered to.

The WebAPI dataset. The *WebAPI* dataset consists of 158,844 tracks from 26,683 unique artists and 74,351 unique albums. Each track contains a 30 second audio clip and a lyrics text file. Each track is associated with one album and one artist. Each album contains a release year and an album art image file, and every artist contains a list of related artists, genre labels, and latitude and longitude coordinates of the artist's originating location. Therefore, every track at the track level has lyrical and raw audio data, at the album level has album cover art and release year data, and at the artist level has related artist, genre, and location data. The data are obtained through various datasets and web application programming interfaces (APIs). Table 1 matches each feature with a list of its sources.

The *WebAPI* dataset is diverse. It contains artists from 153 different countries whose careers span 102 years (between 1917 and 2019) and are labeled with 2347 unique genres. The *WebAPI* dataset is also highly unbalanced. Artists in this dataset originate mostly in the United States (46.16%), the United Kingdom (12.49%), Canada (4.73%), and Australia (2.57%). 52.04% of tracks were released in the past 10 years, 25.82% in the early 2000s, 11.18% in the 1990s, and around 4% each in the 1980s, 1970s, and 1960s. Also, 50% of the artists can by labeled using only 178 genres. Because it is diverse, the *WebAPI* dataset exhibits the *long tail* phenomenon (see Figure 7). This fact will be true for any large catalog of music. Therefore, an unbalanced dataset is desired over a balanced dataset.

The most represented country is the United States, although exploring this dataset by country is only for convenience. Machine learning models will be given more geographically precise information (latitudes and longitudes) to train on. Differences between cities and regions within the same country could be important, especially in larger countries like the United States.

Music of the past can be separated neatly by decade. In the West, music of the 50s, 60s, 70s, 80s, and 90s is usually understood to have its own distinct styles and genres. Most of the music in the dataset was released in the past 20 years, however.

The top occurring genres include variants of 'rock', 'pop', 'metal', 'hip hop', and 'country'. Under-represented genres are geographical genres like 'chinese metal' and 'victoria bc indie', specific sub-genres like 'ambient dub techno' and 'ska jazz', and extremely niche genres like 'ambient psychill', 'breakcore', and 'gothic doom'. Genres are not used to train machine learning models.

The dataset is split into 80% training, 10% validation, and 10% testing sets that are filtered for artists to be represented in one set only. It is also split in a way that maintains equal artist representation ratios of locations, years, and genres. The main motivations behind building the *WebAPI* dataset are to gather data that have a useful ground truth for recommendation (related artists) and to gather data that have reliable metadata (lyrics, album art, release year, location) to test how much influence metadata has on recommendation when used in conjunction with raw audio data. Note that this dataset contains some incorrectly-labeled data due to the imperfect process of scraping the web and various APIs. However, this small amount of noise is negligible and can be ignored.

Cifar-100. The *Cifar-100* dataset (Krizhevsky & Hinton, 2009) consists of small color images divided into 100 classes. The purpose of including an unrelated image dataset is to benchmark musically-motivated CNN architectures against a common CNN architecture that is optimized for image-related tasks.

Features

Spectral features. All audio was sampled at $22050Hz$. This downsampled most of the audio tracks in the *FMA* dataset by half, cutting the subsequent computation and memory requirements for that dataset by half. Listening to the audio and visualizing spectral representations confirmed that most of the information is retained at $22050Hz$. As is common practice in MIR, a compressive nonlinearity was applied to the magnitude of each spectral representation as $X \mapsto 10 \times \log_{10}(X^2)$ to

better approximate the human auditory response. Inputs to deep learning models are normalized to have zero mean and a standard deviation of one.

The STFT parameters were chosen to maximize detail while keeping computation and memory usage reasonably small. The Fast Fourier Transform (FFT) size was chosen to be 4096, using a power of two optimizes the FFT algorithm. The window was chosen to match the FFT size in length and to use a Hann window function with a hop length of 1/4 of the window length. These parameters correspond roughly to a $185ms$ window applied every $46ms$. STFT analysis on a $30s$ clip of audio results in a 2049×643 complex valued tensor that describes the magnitude and phase of the sinusoids present in each time window. The output of the FFT for real-valued signals is symmetric, so only half of the output is maintained. Only the magnitude of the complex value is used.

Some other STFT parameter decisions that were experimented with and considered include using the Blackman-Harris window function, a smaller hop length, and zero-padding the window. The Blackman-Harris window has a larger main-lobe bandwidth than the Hann window and requires a larger FFT size and window length to match the Hann frequency peak detection; because of memory considerations, this was not used. Using a smaller hop length also increased the size of the resultant tensor, and a hop of 1/4 the window seemed a good compromise for most applications. Using a smaller window size than the FFT and zero-padding the window would create a smoother spectrum, but the differences were negligible, so for increased simplicity the window was chosen to always match the FFT size.

As is the case for deep learning, Fourier-based analysis has many tuneable hyperparameters to consider. Larger window lengths improve the resolution of frequency and harmonic content, and smaller window lengths improve the resolution of the attacks and segments. Two separate spectrums could have been produced that favor either time or frequency separately, which then could have been analyzed in parallel with the results then combined. Instead, a good time-frequency compromise was used for all analyses due to time and memory considerations.

Some models were trained using the bottom half of the STFT spectrogram. For

music, lower frequencies are more important. The advantage of mel-scaled STFTs and constant-q transforms is that they reduce the resolution of higher frequencies logarithmically, reducing the dimensionality of the audio representation while maintaining the important information. Cutting the STFT spectrogram in half is a crude way of reducing the dimensionality from $2049 \times time$ to $1024 \times time$, while keeping most of the important spectral information. The mel-scaled spectrograms are created using a mel basis of size 256, reducing the dimensionality further to $256 \times time$. A further reduced representation can be achieved by taking the first 13 MFCCs (discarding the first) resulting in a $12 \times time$ tensor. The musically-motivated constant-q transform is implemented using 24 bins per octave, starting at $C_1(\sim 32.7Hz)$ over seven octaves. The result is a spectrogram of size $168 \times time$. Wrapping this result into 12 pitch classes results in a $12 \times time$ chromagram. Each representation makes trade-offs between size and resolution.

Metadata. All album art is $640px \times 640px$ with $0 - 255$ RGB color channels. Different sized images are scaled to match. Pixel values are compressed into the range $[0, 1]$ by dividing each pixel value by 255 before being used as input to any deep learning model. Latitude, longitude, and year data are all normalized to have zero mean and a standard deviation of one before being used as input into any deep learning model.

Lyrical data requires more pre-processing than other metadata. Lyrics are transformed into lists of integers that map to words. The integer that is one greater than the total number of words used is reserved for infrequent words for some models that don't use the entire vocabulary. The input of lyrics into a model will always first go through an embedding layer that maps an integer into a vector representation of that word. Lyric data is always batched. The integer 0 (not a word) is prepended to a lyric list until all lyric lists in a batch are the same length as the longest lyric list in that batch. The subsampling method, as proposed by Mikolov, Sutskever, Chen, Corrado, and Dean (2013), is used to probabilistically drop words by their frequency in each batch.

Deep Learning Architectures

Convolutional neural networks.

The Time/Frequency Favoring hypothesis. The best-performing CNN architectures on the ImageNet dataset (Russakovsky et al., 2015) and thus some of the most popular and regularly-used architectures for general image-related tasks include ResNet (He, Zhang, Ren, & Sun, 2015), VGGNet (Simonyan & Zisserman, 2014), Inception (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016), and Xception (Chollet, 2017). One thing they have in common is the use of square kernels (usually 3×3) for convolutions and pooling operations. Because of the statistical invariance of objects in images in both the horizontal and vertical axes, square kernels make sense for mapping spacial correlations and are well suited for image-related tasks. Spectrograms, however, do not share the same kind of statistical invariance, so it is assumed that there exists a music domain specific architecture that is more efficient and/or that outperforms the state-of-the-art architectures used for image-related tasks.

Rhythm and melody are horizontal elements of music, and they are described across the time domain. The vertical elements of music include harmony, timbre, and texture, and they lie in the frequency domain. Suppose the spectrogram dimensions are $H \times W$, then a convolution with a kernel of size $1 \times n$ should only be capable of mapping horizontal elements, and a convolution with a kernel of size $n \times 1$ should only be capable of mapping vertical elements. It is hypothesized that forcing a CNN to favor either time or frequency by using wide or tall kernels will allow it to learn richer musically-inspired features more efficiently than a model using square-shaped kernels.

As an initial experiment of this hypothesis, consider an edge-detecting 3×3 kernel with a weight of 8 in the middle and a weight of -1 in each surrounding pixel. Such a kernel will detect edges vertically and horizontally when convolved over an image. Flattening this kernel to shapes 9×1 and 1×9 while keeping the 8 in the middle will create kernels that detect only horizontal and vertical edges, respectively. The results of these convolutions are presented in Figure 8. Note that a square kernel is capable of acting as an exclusive vertical or horizontal edge detector by zeroing out some of the

weights.

A deep model will learn the best representation of the data given enough examples and time. For example, if the horizontal elements happen to be the most representative of the data in a MGR task, the model hypothetically could set the activations to zero in the sides of a square kernel, effectively making it a $n \times 1$ kernel. Most of the CNN architectures in the MIR literature, if even specified, use typical square kernels, and they show good results (Murauer & Specht, 2018; Nanni, Costa, Aguiar, Silla Jr, & Brahnam, 2018). However, some literature suggests that more musically-motivated architectures should be preferred.

X. Li, Li, Fern, and Raich (2016) analysed the cross-correlation between neighboring pixels of all examples in a dataset to derive an optimal kernel shape that when used in a CNN showed improvements over traditional architectures in model generalization capability, robustness to hyperparameter tuning, and classification accuracy. The authors applied the method to natural images (in which they justified the use of square kernels), bioacoustic spectrograms, and gene sequence data. For bioacoustic spectrograms, the correlations appeared strongest in the horizontal and vertical directions. Pons, Lidy, and Serra (2016) propose that using wide and tall CNN kernels will capture more musically-motivated features from spectrograms. Empirical evidence suggests that using a variety of $n \times 1$ kernels (Pons, Slizovskaia, Gong, Gómez, & Serra, 2017) or a variety of $1 \times n$ kernels (Pons & Serra, 2017) in the first layer of a CNN might help it learn musically-relevant features with fewer parameters.

Architectures. To test the validity of favoring time and/or frequency, three CNN architectures are proposed. A fourth architecture is used only on album artwork and is not related to spectrograms. Some experimentation went into deciding on the final architecture designs, including using different activation functions, residual layers, deeper and wider designs, and global max pooling at the output.

Simple. The *Simple* architecture (Figure 9) is inspired by, and is a simplified version of, the Xception architecture. The input first goes through two convolutions with 3×3 kernels and of stride 2 each to reduce dimensionality. This is followed by

depthwise separable convolutions with 3×3 kernels and max pooling layers with 3×3 kernels and strides of 2 until a global average pooling layer. The output of the global average pooling layer is fed into a fully-connected (dense) layer that reduces the output to the number of classes. Unless directly followed by a max pooling layer, each convolution layer is followed by a rectified linear unit (ReLU) activation function (Nair & Hinton, 2010). A ReLU activation function is applied after each max pooling layer as well. Unlike the Xception architecture, no residual layers are used in this simple version. As in the Xception architecture, batch normalization layers follow every convolutional layer.

Time. In the *Time* architecture (Figure 10), six different convolutional layers are applied to the input and concatenated before following the same pattern as in the *Simple* architecture. The six input convolutions have different $1 \times n$ kernels. The varying value for n is meant to capture different musical concepts. A large n should be able to detect rhythm or tempo, and a small n should be able to more efficiently detect onsets and attacks. Like the *Simple* architecture, the concatenated input follows layers of depthwise separable convolutions and pooling layers with batch normalization and ReLU activation functions. Unlike the *Simple* architecture, the convolutional layers have 1×9 kernels, and the max pooling layers are meant to reduce the dimension of frequency with 2×1 kernels and 2×1 strides. Also, a convolution with valid padding and a kernel of $F \times 1$, where F is the height dimension of the input, the output of the previous layer, is used to flatten the frequency dimension to 1 before the global average pooling layer.

Freq. The *Freq* architecture (Figure 11) is exactly the same as the *Time* architecture but with reversed kernels, max pooling layers that reduce the dimension of time with 1×2 kernels and 1×2 stride, and a flattening of time before the global average pooling layer at the output instead of frequency.

Albums. The CNN trained on album artwork is a copy of the MobileNetV2 architecture, a small but powerful CNN architecture for image tasks (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018).

Recurrent neural networks. RNNs are designed to model data that contains temporal dependencies by incorporating memory into their construction. RNNs are already well-suited for sequential data like audio and text, so little experimentation went into the construction of their architectures. Both Long Short-Term Memory Cells (LSTMs) (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Units (GRUs) (Cho et al., 2014) are used.

Architectures.

RNN. The *RNN* architecture (Figure 12) is a stack of 3 GRU layers, all with an output dimension of 256 fed into a final dense layer at the output.

LargeRNN. The *LargeRNN* architecture (Figure 13) is a stack of 3 LSTM layers with increasing output dimensions of 256, 512, and 1024 fed into a final dense layer at the output.

Lyrics. The *Lyrics* architecture (Figure 14) is the same as the *LargeRNN* architecture, but it contains an embedding layer at the input that transforms an input word into a 200 dimensional vector before being fed into the RNN. The embedding layer in this architecture is not pre-trained. It is learned during training on lyrical data.

Ensembles.

Architectures.

TimeFreq. The *TimeFreq* architecture (Figure 15) is a stacked ensemble of the trained *Time* and *Freq* models that takes as input the logits from both trained architectures and feeds them into a neural network consisting of dense layers with ReLU activation functions. This architecture uses 50% dropout and 0.002 L2 normalization to combat overfitting.

TimeFreqAvg. *TimeFreqAvg* is simply the mean of the *Time* and *Freq* architecture outputs.

GenreEnsemble. The *GenreEnsemble* architecture (Figure 15) combines the output logits of the *Time* and *Freq* models trained on halved STFT spectrograms, mel-scaled STFT spectrograms, and constant-q transforms with two *RNN* models trained on chromagrams and MFCCs. All inputs are concatenated and fed into a neural network consisting of dense layers with ReLU activation functions. This architecture

uses 50% dropout and 0.002 L2 normalization to combat overfitting. The *GenreEnsemble* architecture is specialized for and only trained on the medium subset of the *FMA* dataset for the purpose of achieving the highest possible accuracy in classifying 16 genres.

Recommendation. The *Recommendation* model architecture (Figure 17) is a stacked ensemble that uses the outputs of all models trained on the *WebAPI* dataset as inputs. No models are trained on the year or location data, so these data are used directly in the *Recommendation* architecture. The architecture consists of dense layers with ReLU activation functions. The *Recommendation* architecture is specialized for and only trained on the *WebAPI* dataset.

Different combinations of inputs can be used to train varying versions of the *Recommendation* architecture. Five different versions are trained in order to make comparisons: the *RecommendationByLyrics*, *RecommendationByAlbums*, *RecommendationByYear*, and *RecommendationByLocation* architectures, which use only the audio-based model outputs concatenated with the output of the *Lyrics* architecture, the output of the *Albums* architecture, with years as input, or with location as input, respectively. The final architecture, the *RecommendationByAudioOnly* architecture, uses only the outputs of the six audio-based models.

Experimental Design

The two datasets used in this study serve different purposes. The *FMA* was used to test audio-based models for genre recognition, and the *WebAPI* dataset was used as the basis for the proposed *Recommendation* model. The *WebAPI* dataset was also used to investigate the effects metadata has on recommendations. Both datasets were used to test The Time/Freq Favoring Hypothesis. All models were trained by optimizing some loss function, depending on the task, with the *Adam* method matching the hyperparameter values as suggested by Kingma and Ba (2014). The learning rate, however, differed depending on the model. All models were trained with small batch sizes of either 6, 8, 16, or 32, depending on the model. Every model was trained using early stopping to avoid overfitting, in which after each epoch, a validation metric was

calculated using the validation set, if it did not improve for some number epochs training stopped and the best model was saved. Most models were trained once with the same learning rate but a few were trained a second or third time with decreasing learning rates. These training strategies were the result of careful experimentation, but they are not ideal due to the infinite scope of tuneable parameters.

Single genre recognition. The medium subset of the *FMA* dataset contains $25,000$ tracks each with a unique genre label and a 30 second audio file, 24 of which were found to be faulty and were removed from the set. The single genre recognition task on the medium subset of the *FMA* dataset is a multiclass classification problem with unbalanced labels (Table 2). An $F1_{micro}$ score was used as the figure of merit on this task and is defined by Equation 1.

Given that TP_g, FP_g, and FN_g are the numbers of true positives, false positives, and false negatives for the given genre g, the $F1_{micro}$ score is calculated as:

$$P_{micro} = \frac{\sum_{g=0}^{16} TP_g}{\sum_{g=0}^{16} TP_g + \sum_{g=0}^{16} FP_g} \tag{1a}$$

$$R_{micro} = \frac{\sum_{g=0}^{16} TP_g}{\sum_{g=0}^{16} TP_g + \sum_{g=0}^{16} FN_g} \tag{1b}$$

$$F1_{micro} = 2 \times \frac{P_{micro} \times R_{micro}}{P_{micro} + R_{micro}} \tag{1c}$$

For multiclass classification, the last layer of each network is a softmax activation function that calculates the probability that the input data belongs to each label, resulting in a 16 dimensional vector representing each of the 16 unique genres. Categorical cross entropy between this vector and a vector of true probabilities (one in the direction of its true label and zeros elsewhere) is used as the loss function.

Multiple genre recognition. The large subset of the *FMA* dataset contains $106,574$ tracks, each with multiple genre labels and a 30 second audio file, $2,290$ of which were found to be faulty and were removed from the set. The multiple genre recognition task on the large subset of the *FMA* dataset is a multi-label classification problem with unbalanced labels. Each track is labeled with any number of the 161 unique genres. Like the medium subset, this dataset is unbalanced. Two figures of merit

are used to evaluate the performance. The first is mean squared error,

$$MSE = \frac{1}{161}\sum_{i=1}^{161}(\vec{y}_i - \vec{x}_i)^2 \tag{2}$$

given the model output in logits as $\vec{x}$ and the true labels as $\vec{y}$. The second is Hamming loss,

$$HL = \frac{1}{161}\sum_{i=1}^{161}\vec{y}_i \otimes \vec{x}_i \tag{3}$$

given the model output predictions as $\vec{x}$, the true labels as $\vec{y}$, and the exclusive-or function donated as $\otimes$. The mean of these two metrics is used over multiple tracks.

For multi-label classification, the last layer of each network is a sigmoid activation function which pushes each logit (each label) toward either zero or one. The resulting 161 dimensional vector represents the probability that the input data should be labeled with any of the 161 unique genres. In this case, 0.5 is used as the threshold; rounding the vector along each dimension results in a vector with a one in the direction of each predicted label and zeros elsewhere. Binary cross entropy calculated individually on every label is used as the loss function.

Content-based recommendation. The *WebAPI* dataset contains 158,844 tracks from 26,683 artists. Every artist is related to up to 20 other artists. The assumption is made that artist relationships are the ground truth for recommendation. In this case, if a user likes artist A and artist A is related to artist B, then the probability that the user likes artist B is higher than the probability that the user likes an artist selected at random. Also, if artist C is related to artist B but not artist A, then the probability that the user likes artist C is higher than the probability that the user likes an artist selected at random but lower than the probability that the user likes artist B. Artist relationships are also assumed to be free from the effects of the *long tail*; niche artists are related to niche artists and are not always related to the most similar popular artists. The last assumption is that artist relationships are based not just on genre and audio content, but on contextual, emotional, cultural, and historical factors as well. For example, a 'christian metal' band may sound very similar to a 'death metal' band, but they should never be related. This example is a counter-argument to the claim that machine learning models can fully categorize music on raw audio alone.

These assumptions are made after extensive exploration of the data and consideration of their source. Artist relationships are taken from the Spotify API. Spotify is the largest music streaming service in the world by number of paid listeners (Wikipedia contributors, 2019). As discussed earlier, Spotify analyses users' listening behavior and playlist generation (collaborative filtering) and deploys deep learning models on different music features, along with scraping the web (content-based), to develop recommendations and artist relationships (Ciocca, 2017). This ongoing effort results in comprehensive artist relationship mapping that can be transformed into a N dimensional embedding (latent) space that preserves relationships through some vector similarity metric (Hoff, Raftery, & Handcock, 2002). With an embedding space, a content-based recommendation model can make recommendations on tracks and artists that do not exist in the catalog by mapping new data onto the embedding space and returning its nearest neighbors. Additionally, this type of model is not subject to the cold start problem because it is able to directly position a new artist into a relevant position in a music catalog or aid a human in doing so.

Training artist embeddings. To train artist embeddings, all artists included in the *WebAPI* dataset and who appear as related artists are first mapped onto $109,883$ indexes. Each index corresponds to a row in an initially randomly-weighted embedding table. The embeddings are trained like every other model in this study, using the *Adam* optimization algorithm. The inputs to the model are the index of an artist and the index of a context artist which are mapped by the embedding table into embedding vectors. The similarity of the two embedding vectors are calculated using cosine similarity,

$$similarity = cos(\theta) = \frac{\vec{e}_{artist} \cdot \vec{e}_{context}}{\|\vec{e}_{artist}\|\|\vec{e}_{context}\|} \tag{4}$$

which ranges between -1 (exactly opposite) to 1 (exactly similar). The cosine similarity metric is then fed into a sigmoid function at the output. This dataflow is visualized in Figure 18. The context artist is either a related artist or a random artist. Related artists have a desired label of 1 and random artists have a desired label of 0. Binary cross entropy is used as the loss function. As the model trains, it will update the

weights of the embedding table so that similar artists will have high similarity, pushing the output to 1, and random artists will have low similarity, pushing the output to 0. It is possible that a related or closely-related artist is picked as a random artist but the effects of this is negligible over many training examples and epochs.

Regression. Mapping input data to an embedding space is a machine learning regression task. For each model, the output activation function is removed. The resulting output vector of logits is compared with the artist embedding that it is trying to predict, with negative cosine similarity as a loss function. For every track, each model trains on that track's artist embedding. Recommendations can be made at the track level or at the album or artist level by averaging outputs.

Every model is trained on the *WebAPI* dataset. Once trained, the output inferences on the following models are saved and used as input to the *Recommendation* model: the *Time* and *Freq* models on mel-scaled STFTs and constant-q spectrograms, the *LargeRNN* models on chromagrams and MFCCs, the *Albums* model on album cover art, and the *Lyrics* model on lyrics. The *Recommendation* model also receives as direct input both location and year data.

Results

Single Genre Recognition

The results of the single genre recognition task for each CNN-based model is outlined in Table 3. The $F1_{micro}$ scores are too close to make any significant conclusions, but a few interesting observations can be made. Firstly, the *Time* architecture actually outperformed the *Simple* architecture in most cases. Secondly, the *TimeFreq* architecture had the highest $F1_{micro}$ for every spectral representation. One should note that the *Time*, *Freq*, and *TimeFreq* architectures are much smaller by number of parameters than the *Simple* architecture.

As a baseline test for validating the *Simple* architecture, each CNN-based model was trained on the *Cifar-100* image dataset, the results of which are presented in Table 4. The *Simple* architecture achieved the highest $F1_{micro}$ score by a large margin on this dataset. This validates the strength of the *Simple* architecture on image data and proves that the architecture's poor performance on spectrograms isn't due to a flaw in the architecture itself or a bug in the code. The *Time* and *Freq* architectures performed similarly. Interestingly, the *TimeFreq* architecture failed to learn anything and made predictions at random. This suggests that the *Time* and *Freq* architectures are learning conflicting features on images but complimentary features on spectrograms.

An attempt was made to achieve the highest $F1_{micro}$ score possible using the *GenreEnsemble* architecture. The resultant model combines the output logits of the *Time* and *Freq* models, which were trained on halved STFT spectrograms, mel-scaled STFT spectrograms, and constant-q transforms, with two *RNN* models that were trained on chromagrams and MFCCs. The *RNN* models trained on chromagrams and MFCCs achieved $F1_{micro}$ scores of 0.467522 and 0.516142 respectively (both models contain $1,000,976$ total parameters). These are slightly lower scores than the CNN-based counterparts, but they were included in the *GenreEnsemble* because they might be capable of learning different temporal features that the CNN-based models miss. The *GenreEnsemble* model achieved an $F1_{micro}$ score of 0.654998 with $3,157,406$ total parameters.

Multiple Genre Recognition

The results of the multiple genre recognition task for each CNN-based model is outlined in Table 5. These results match what was observed in the single genre recognition task: the *Time* architecture outperformed the *Simple* architecture in all cases, and the *TimeFreq* architecture had the lowest mean squared error and Hamming loss for every spectral representation. Averaging the outputs of the *Time* and *Freq* models trained on halved STFT spectrograms, mel-scaled STFT spectrograms, and constant-q transforms results in a mean squared error of 0.014926 and a Hamming loss of 0.018012.

Content-Based Recommendation

The loss function and figure of merit for any model training on artist embeddings is cosine similarity. The higher the similarity, the better the recommendations. However, this metric is only as good as the ground truth, which in this case is the artist's embeddings themselves. To get a sense of how well the embedding model is able to map artists into vectors, a comparison is made between an artist's related artists and its nearest neighbors in the embedding space. This requires two equally sized sets, $A_{related}$ and $A_{embeddimg}$ where $A_{embeddimg}$ is the N nearest neighbors of an artist in embedding space where $N = |A_{related}|$. For most artists, $|A_{related}| = 20$. These sets are compared using the Jaccard similarity coefficient, a set similarity metric defined as the size of the intersection divided by the size of the union of the two sets. The average Jaccard similarity coefficient over every artist in the embedding space was equal to 0.548842. This means with some certainty that all artists most likely maintained at least 50% of their related artists as nearest neighbors in the embedding space.

Another way to interpret the embedding space is by using T-distributed Stochastic Neighbor Embedding (t-SNE) (Maaten & Hinton, 2008) to reduce the embedding space to two dimensions for visualization. Clear clusters are visible when visualizing the t-SNE embeddings by country (Figure 19) and genre (Figure 20). Clusters are less clear when visualizing the t-SNE embeddings by year (Figure 21). This is likely due to the way years are calculated, which is by taking the average of the

artist's tracks release years. Plotting a few distinct artists with two levels of related artists results in the closest clustering (Figure 22). Some expected patterns can be seen in each plot. For example, at around $-40, -40$ Jamaican reggae music is observed in the country, genre, and artist plots.

The exploration of the embedding space suggests that it is a satisfactory ground truth for training deep learning models. Therefore, the cosine similarity between the true artist embedding and a model's predicted artist embedding has a correlation with good recommendations and proper placements in the embedding space. The results of this regression task are presented in Table 6. The validity of The Time/Freq Favoring Hypothesis is further strengthened by these results. Again, the *TimeFreqAVG* architecture performed better, and with fewer total parameters, than did the *Simple* architecture. The *Lyrics* model, when compared to audio-based models, performed about the same. The *Albums* model on its own did not perform well. However, the *RecommendationByAlbums* architecture slightly outperformed the *RecommendationByAudioOnly* architecture, meaning that there might be some correlation between album artwork and recommendation; however, more experimentation here is needed. Both lyrics and location data were able to greatly improve the audio-based model when combined.

As expected, the *Recommendation* architecture attained the highest average cosine similarity on the testing set. This trained model was used to make recommendations for artists in the testing set by averaging the embedding predictions over every track available from the artist. A few of the recommendations made by the model are shown in Table 7. Trained on the *WebAPI* dataset, this model did well recognizing music that is distinct in genre like 'contemporary christian' bands, music that is highly regional, like 'OPM' in the Philippines, and music unique in sound relative to the rest of the catalog, like 'cathedral choir' music. This model was able to successfully place conforming music into broad groups but had trouble with more unique music. For example, the Vermont 'jam band' Phish, known for their distinct style and blending of genres, should be recommended along with other unique 'jam

bands' like Tea Leaf Green or the Disco Biscuits. Instead, this model puts Phish next to 'alternative' acts like Cake and 'folk-pop' artists like Pete Yorn. Whether or not these are related artists is a matter of opinion, but using the related artist data as a ground truth reveals that these bands are a separated by few degrees; relative to the rest of the catalog however, they are still close. In fact, after scrutinizing many recommendations at random, it is arguable that the model never makes obvious mistakes. Even seemingly unrelated artists like Ravi Shankar, a composer of Hindustani classical music, and the Irish family band Claanad, which the model recommends, share some musical characteristics. In this case, the model might be finding similarities between the sitar and the mandolin. However, more data would likely improve the model.

Conclusion

The proposed content-based recommendation model is not entirely content-based. The model is trained on a ground truth built using both collaborative and content-based methods. This might make it a 'hybrid' model since it depends on collaborative filtering to develop the artist embedding space. Regardless, the model does use only content-based inputs. This means that it is not affected by the cold start problem, which makes it useful in a production environment that can not handle the scale of manual new music introduction. Touching on an earlier example, this is the problem Pandora faces with The Music Genome Project, which is essentially a human-powered embedding model (Howe, 2009). Carefully but manually labeling a particular piece of music with 400 descriptive 'genes' by domain experts is the same concept behind training a machine learning model to produce artist embeddings. In fact, the embedding dimension of 800 was chosen to show how easy it would be for a machine learning model to double the dimension of the music vector representation produced by The Music Genome Project. The Music Genome Project is a good idea, but its implementation does not scale.

One way of improving the model would be to improve the artist embeddings. Cosine similarity was used because of its simplicity. However, cosine similarity is a directional similarity metric. A distance metric that considers both direction and magnitude, like euclidean distance, might provide improved embeddings. Also, using embeddings at the track level is possible and might improve the resolution of recommendations. Another possible route is to create multiple models and multiple embedding spaces that increase in resolution. The proposed content-based recommendation model is trained on a very diverse dataset, and it is able to map artists in the 'general' direction. A general model, like the one proposed, could point to a more specific model that is trained on a less diverse area of consideration. Improving the embedding space and experimenting with different model resolutions are worthwhile research topics.

The *Lyrics* architecture was on par with the audio-based architectures, even

though little experimentation or consideration went into the construction of the architecture. However, the lyrics data are mostly English, so it is possible that the *Lyrics* architecture performed well because it could distinguish between different languages, not because it was understanding the meaning behind the words. A more balanced dataset could provide clearer results in this regard. Improving the *Lyrics* architecture is one worthy direction of future research, especially with recent breakthroughs in natural language processing and understanding. Lyrics could be the most representative feature for content-based recommendation. Recommendation on album artwork, however, does not seem to be a useful pursuit.

The imbalance of the dataset was shown most clearly when considering the year feature. The performance of the *RecommendationByYear* architecture was much worse than expected. The earlier one goes back in time, the more likely it is that artists of that era are related because of the lack of representation. Unfortunately, data was sparse for artists of the 40s, 50s, 60s, 70s, and 80s. The dataset also lated full track lengths. Possible avenues such as song segmentation could not be explored because the audio data was limited to 30 second clips. Access to full length tracks would likely improve a content-based recommendation model.

Although the proposed content-based recommendation model is returning similar artists by genre and origin, the quality of the recommendations is ultimately unknown. The assumption was made that related artists are the ground truth for recommendation, and the model was evaluated accordingly. However, the value of each recommendation is ultimately up to the consumer, and this study lacked 'online' evaluation methods like A/B testing and surveys. Despite this, however, a few arguments can still be made. Firstly, strong experimental evidence suggests the validity of both The Time/Frequency Favoring Hypothesis and the hypothesis that raw audio data are not representative enough for machine learning models to make acceptable recommendations. Secondly, for every dataset and task, the musically-motivated *TimeFreq* architecture outperformed the computer vision optimized *Simple* architecture. Finally, the addition of metadata strengthened every model.

References

Anderson, C. (2004). The long tail. *Wired magazine*, *12*(10), 170–177.

Aucouturier, J.-J., & Pachet, F. (2002). Music similarity measures: What's the use? In *Ismir* (pp. 13–17).

Aucouturier, J.-J., Pachet, F., & Sandler, M. (2005). "the way it sounds": timbre models for analysis and retrieval of music signals. *IEEE Transactions on Multimedia*, *7*(6), 1028–1035.

Benetos, E., & Kotropoulos, C. (2008). A tensor-based approach for automatic music genre classification. In *Signal processing conference, 2008 16th european* (pp. 1–4).

Berenzweig, A., Logan, B., Ellis, D. P., & Whitman, B. (2004). A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, *28*(2), 63–76.

Bergstra, J., Casagrande, N., Erhan, D., Eck, D., & Kégl, B. (2006). Aggregate features and adaboost for music classification. *Machine learning*, *65*(2-3), 473–484.

Bertin-Mahieux, T., Eck, D., Maillet, F., & Lamere, P. (2008). Autotagger: A model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, *37*(2), 115–135.

Bou-Rabee, A., Go, K., & Mohan, K. (2012). *Classifying the subjective: Determining genre of music from lyrics.* Citeseer.

Brown, J. C. (1991). Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, *89*(1), 425–434.

Chen, S., Moore, J. L., Turnbull, D., & Joachims, T. (2012). Playlist prediction via metric embedding. In *Proceedings of the 18th acm sigkdd international conference on knowledge discovery and data mining* (pp. 714–722).

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, *abs/1406.1078*. Retrieved from `http://arxiv.org/abs/1406.1078`

Choi, K., Fazekas, G., Cho, K., & Sandler, M. (2017). A tutorial on deep learning for

music information retrieval. *arXiv preprint arXiv:1709.04396*.

Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). Convolutional recurrent neural networks for music classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 2392–2396).

Chollet, F. (2017, July). Xception: Deep learning with depthwise separable convolutions. In *The ieee conference on computer vision and pattern recognition (cvpr).*

Ciocca, S. (2017, October). *Spotify's discover weekly: How machine learning finds your new music.* Retrieved from `https://medium.com/s/story/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe` ([Online; posted 10-October-2017])

Clendinning, J. P., & Marvin, E. W. (2016). *The musician's guide to theory and analysis*. WW Norton & Company.

Costa, Y. M., Oliveira, L. S., & Silla Jr, C. N. (2017). An evaluation of convolutional neural networks for music classification using spectrograms. *Applied soft computing*, *52*, 28–38.

Defferrard, M., Benzi, K., Vandergheynst, P., & Bresson, X. (2017). Fma: A dataset for music analysis. In *18th international society for music information retrieval conference.* Retrieved from `https://arxiv.org/abs/1612.01840`

Dieleman, S., & Schrauwen, B. (2014). End-to-end learning for music audio. In *Acoustics, speech and signal processing (icassp), 2014 ieee international conference on* (pp. 6964–6968).

Flexer, A. (2007). A closer look on artist filters for musical genre classification. *World*, *19*(122), 16–17.

Flexer, A., Schnitzer, D., Gasser, M., & Widmer, G. (2008). Playlist generation using start and end songs. In *Ismir* (Vol. 8, pp. 173–178).

Font, F., Roma, G., & Serra, X. (2013, 21/10/2013). Freesound technical demo. In *Acm international conference on multimedia (mm'13)* (p. 411-412). Barcelona, Spain: ACM. doi: 10.1145/2502081.2502245

Fule, P., & Roddick, J. F. (2004). Detecting privacy and ethical sensitivity in data mining results. In *Proceedings of the 27th australasian conference on computer science-volume 26* (pp. 159–166).

Gjerdingen, R. O., & Perrott, D. (2008). Scanning the dial: The rapid recognition of music genres. *Journal of New Music Research*, *37*(2), 93–100.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*. Retrieved from `http://arxiv.org/abs/1512.03385`

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hoff, P. D., Raftery, A. E., & Handcock, M. S. (2002). Latent space approaches to social network analysis. *Journal of the american Statistical association*, *97*(460), 1090–1098.

Holzapfel, A., & Stylianou, Y. (2008). Musical genre classification using nonnegative matrix factorization-based features. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(2), 424–434.

Howe, M. (2009). *Pandora's music recommender.* A Case Study, I,.

Jeong, I.-Y., & Lee, K. (2016). Learning temporal features using a deep neural network and its application to music genre classification. In *Ismir* (pp. 434–440).

Jiang, D.-N., Lu, L., Zhang, H.-J., Tao, J.-H., & Cai, L.-H. (2002). Music type classification by spectral contrast feature. In *Multimedia and expo, 2002. icme'02. proceedings. 2002 ieee international conference on* (Vol. 1, pp. 113–116).

Karpov, I., & Subramanian, D. (2002). Hidden markov classification for musical genres. *Course Project*.

Katarya, R., & Verma, O. P. (2018). Efficient music recommender system using context graph and particle swarm. *Multimedia Tools and Applications*, *77*(2), 2673–2687.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*. Retrieved from `http://arxiv.org/abs/1412.6980`

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny*

images (Tech. Rep.). Citeseer.

Laurier, C., Herrera, P., Mandel, M., & Ellis, D. (2007). Audio music mood classification using support vector machine. *MIREX task on Audio Mood Classification*, 2–4.

Lee, C.-H., Shih, J.-L., Yu, K.-M., & Su, J.-M. (2007). Automatic music genre classification using modulation spectral contrast feature. In *Multimedia and expo, 2007 ieee international conference on* (pp. 204–207).

Li, T., & Ogihara, M. (2004). Content-based music similarity search and emotion detection. In *Acoustics, speech, and signal processing, 2004. proceedings. (icassp'04). ieee international conference on* (Vol. 5, pp. V–705).

Li, T., Ogihara, M., & Li, Q. (2003). A comparative study on content-based music genre classification. In *Proceedings of the 26th annual international acm sigir conference on research and development in information retrieval* (pp. 282–289).

Li, T. L., & Chan, A. B. (2011). Genre classification and the invariance of mfcc features to key and tempo. In *International conference on multimedia modeling* (pp. 317–327).

Li, X., Li, F., Fern, X., & Raich, R. (2016). Filter shaping for convolutional neural networks.

Lidy, T., & Rauber, A. (2005). Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Ismir* (pp. 34–41).

Lidy, T., Rauber, A., Pertusa, A., & Quereda, J. M. I. (2007). Improving genre classification by combination of audio and symbolic descriptors using a transcription systems. In *Ismir* (pp. 61–66).

Logan, B. (2000). Mel frequency cepstral coefficients for music modeling. In *Ismir* (Vol. 270, pp. 1–11).

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Maillet, F., Eck, D., Desjardins, G., & Lamere, P. (2009). Steerable playlist generation by learning song similarity from radio station playlists. In *Ismir* (pp. 345–350).

Mandel, M. I., & Ellis, D. (2005). Song-level features and support vector machines for music classification. In *Ismir* (Vol. 2005, pp. 594–599).

Mayer, R., Neumayer, R., & Rauber, A. (2008). Rhyme and style features for musical genre classification by song lyrics. In *Ismir* (pp. 337–342).

McKinney, M., & Breebaart, J. (2003). Features for audio and music classification.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, *abs/1310.4546*. Retrieved from `http://arxiv.org/abs/1310.4546`

Müller, M., Kurth, F., & Clausen, M. (2005). Audio matching via chroma-based statistical features. In *Ismir* (Vol. 2005, p. 6th).

Murauer, B., & Specht, G. (2018). Detecting music genre using extreme gradient boosting. In *Companion proceedings of the the web conference 2018* (pp. 1923–1927). Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee. Retrieved from `https://doi.org/10.1145/3184558.3191822` doi: 10.1145/3184558.3191822

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 807–814).

Nanni, L., Costa, Y. M., Aguiar, R. L., Silla Jr, C. N., & Brahnam, S. (2018). Ensemble of deep learning, visual and acoustic features for music genre classification. *Journal of New Music Research*, 1–15.

Oord, A. V. d., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in neural information processing systems* (pp. 2643–2651).

Oramas, S., Nieto, O., Barbieri, F., & Serra, X. (2017). Multi-label music genre classification from audio, text, and images using deep features. *arXiv preprint arXiv:1707.04916*.

Pampalk, E. (2005). Speeding up music similarity. *2nd Annual Music Information Retrieval eXchange, London*.

Pampalk, E., Flexer, A., & Widmer, G. (2005). Improvements of audio-based music similarity and genre classificaton. In *Ismir* (Vol. 5, pp. 634–637).

Panagakis, I., Benetos, E., & Kotropoulos, C. (2008). Music genre classification: A multilinear approach. In *Ismir* (pp. 583–588).

Patch, N. (2016, January). *Meet the man classifying every genre of music on spotify–all 1,387 of them.* Retrieved from `https://www.thestar.com/entertainment/2016/01/14/meet-the-man-classifying-every-genre-of-music-on-spotify-all-1387-of-them.html` ([Online; posted 14-January-2016])

Peng, W., Li, T., & Ogihara, M. (2007). Music clustering with constraints. In *Ismir* (pp. 27–32).

Pilhofer, M., & Day, H. (2015). *Music theory for dummies.* John Wiley & Sons.

Pons, J., Lidy, T., & Serra, X. (2016, June 15-17). Experimenting with musically motivated convolutional neural networks. In *Proceedings of the 14th international workshop on content-based multimedia indexing (cbmi 2016).* Bucharest, Romania. Retrieved from `http://jordipons.me/media/CBMI16.pdf` doi: 10.1109/CBMI.2016.7500246

Pons, J., & Serra, X. (2017). Designing efficient architectures for modeling temporal features with convolutional neural networks. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 2472–2476).

Pons, J., Slizovskaia, O., Gong, R., Gómez, E., & Serra, X. (2017). Timbre analysis of music audio signals with convolutional neural networks. *CoRR*, *abs/1703.06697*. Retrieved from `http://arxiv.org/abs/1703.06697`

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015, Dec 01). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252. doi: 10.1007/s11263-015-0816-y

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., & Chen, L. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, *abs/1801.04381*. Retrieved from

http://arxiv.org/abs/1801.04381

Schmidt-Jones, C. (2013). *Understanding basic music theory*. Rice University.

Schörkhuber, C., & Klapuri, A. (2010). Constant-q transform toolbox for music processing. In *7th sound and music computing conference, barcelona, spain* (pp. 3–64).

Shao, X., Xu, C., & Kankanhalli, M. S. (2004). Unsupervised classification of music genre using hidden markov model. In *Multimedia and expo, 2004. icme'04. 2004 ieee international conference on* (Vol. 3, pp. 2023–2026).

Sigtia, S., & Dixon, S. (2014). Improved music feature learning with deep neural networks. In *2014 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 6959–6963).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, *abs/1409.1556*. Retrieved from http://arxiv.org/abs/1409.1556

Stevens, S. S., Volkmann, J., & Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, *8*(3), 185–190.

Sturm, B. L. (2012a). An analysis of the gtzan music genre dataset. In *Proceedings of the second international acm workshop on music information retrieval with user-centered and multimodal strategies* (pp. 7–12).

Sturm, B. L. (2012b). A survey of evaluation in music genre recognition. In *International workshop on adaptive multimedia retrieval* (pp. 29–66).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016, June). Rethinking the inception architecture for computer vision. In *The ieee conference on computer vision and pattern recognition (cvpr)*.

Thickstun, J., Harchaoui, Z., & Kakade, S. (2016). Learning features of music from scratch. *arXiv preprint arXiv:1611.09827*.

Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, *10*(5), 293–302.

Van Wel, L., & Royakkers, L. (2004). Ethical issues in web data mining. *Ethics and Information Technology*, *6*(2), 129–140.

Wang, S. (2016). *Musical genre categorization using support vector machines.* Np.

Wikipedia contributors. (2019). *Comparison of on-demand music streaming services — Wikipedia, the free encyclopedia.* `https://en.wikipedia.org/w/index.php?title=Comparison_of_on-demand_music_streaming_services&oldid=893992353`. ([Online; accessed 6-May-2019])

Table 1

WebAPI dataset features and sources

Feature	Sources
Artist Name	Spotify[1]
Artist Origin	MusicBrainz[2], simplemaps[3], Wikipedia[4], Google Maps[5], MSD Artist Locations Dataset[6], Musixmatch[7]
Artist Genres	Spotify
Related Artists	Spotify
Album Name	Spotify
Album Art	Spotify
Album Year	Spotify
Track Name	Spotify
Track Audio	Spotify
Track Lyrics	Musixmatch, Genius[8], AZLyrics[9], LyricWiki[10], Kaggle Dataset 1[11], Kaggle Dataset 2[12], Kaggle Dataset 3[13], Kaggle Dataset 4[14]

[1] https://developer.spotify.com/documentation/web-api/

[2] https://musicbrainz.org/doc/Development/XML_Web_Service/Version_2

[3] https://simplemaps.com/data/world-cities

[4] https://www.mediawiki.org/wiki/API:Main_page

[5] https://developers.google.com/maps/documentation/

[6] https://labrosa.ee.columbia.edu/millionsong/sites/default/files/AdditionalFiles/artist_location.txt

[7] https://developer.musixmatch.com/documentation

[8] https://docs.genius.com/

[9] https://www.azlyrics.com/

[10] https://github.com/rhnvrm/lyric-api

[11] https://www.kaggle.com/rakannimer/billboard-lyrics

[12] https://www.kaggle.com/artimous/every-song-you-have-heard-almost

[13] https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics

[14] https://www.kaggle.com/mousehead/songlyrics

Table 2

FMA dataset medium subset splits

Label	Training	Validation	Testing	Label	Training	Validation	Testing
Rock	5676	711	710	Classical	495	62	62
Electronic	5048	631	632	Old-Time / Historic	408	51	51
Experimental	1799	225	224	Jazz	306	39	42
Hip-Hop	1752	220	220	Country	142	18	39
Folk	1214	152	174	Spoken	94	18	18
Instrumental	1043	131	152	Soul-RnB	94	12	12
Pop	945	122	119	Blues	58	8	8
International	814	102	102	Easy Listening	13	2	6

Note. This dataset is highly unbalanced but the ratios of tracks in each set per genre is consistent.

Table 3

CNN-based single genre recognition results

	Results		
Spectral Representation	Architecture	$F1_{micro}$	Total Parameters
STFT	*TimeFreq*	0.627382	190,510
	Time	0.612991	84,500
	Simple	0.611046	733,736
	Freq	0.585375	62,602
STFT Halved	*TimeFreq*	0.630494	173,366
	Time	0.597822	67,988
	Simple	0.595488	733,736
	Freq	0.582264	61,970
Mel-Scaled STFT	*TimeFreq*	0.627771	160,682
	Time	0.619992	55,700
	Simple	0.609102	733,736
	Freq	0.584986	61,574
Constant-Q	*TimeFreq*	0.619214	159,294
	Time	0.615714	54,292
	Simple	0.584208	733,736
	Freq	0.544535	61,594
MFCCs	*TimeFreq*	0.550758	157,200
	Simple	0.549592	733,736
	Time	0.549203	53,140
	Freq	0.506807	60,652
Chromagram	*TimeFreq*	0.536756	157,200
	Time	0.495138	53,140
	Simple	0.492804	733,736
	Freq	0.478802	60,652

Table 4

Cifar-100 multiclass classification results

Architecture	$F1_{micro}$	Total Parameters
Simple	0.5708	819,980
Freq	0.4684	73,288
Time	0.4540	73,288
TimeFreq	0.0100	243,828

Table 5

CNN-based multiple genre recognition results

	Results			
Spectral Representation	Architecture	Hamming Loss	Mean Squared Error	Total Parameters
STFT Halved	*TimeFreqAVG*	0.018139	0.015105	204, 488
	Time	0.018441	0.015461	105, 253
	Simple	0.018578	0.015624	882, 361
	Freq	0.018678	0.015580	99, 235
Mel-Scaled STFT	*TimeFreqAVG*	0.018142	0.015101	190, 416
	Time	0.018464	0.015377	92, 965
	Simple	0.018489	0.015461	882, 361
	Freq	0.018728	0.015674	98, 839
Constant-Q	*TimeFreqAVG*	0.018349	0.015314	190, 416
	Time	0.018475	0.015602	91, 557
	Simple	0.018681	0.015716	882, 361
	Freq	0.018682	0.015684	98, 859

Table 6

Embedding regression results

		Results	
Features	Architecture	Cosine Similarity	Total Parameters
Mel-Scaled STFT	*TimeFreqAVG*	0.4043262	520, 250
	Time	0.4025255	257, 188
	Simple	0.3799719	1, 537, 336
	Freq	0.3306473	263, 062
Constant-Q	*TimeFreqAVG*	0.3937517	518, 862
	Freq	0.3791033	263, 082
	Simple	0.3787154	1, 537, 336
	Time	0.3503861	255, 780
MFCCs	*LargeRNN*	0.3105006	8, 973, 088
Chromagram	*LargeRNN*	0.2704273	8, 973, 088
Lyrics	*Lyrics*	0.3660617	108, 755, 200
Album Artwork	*Albums*	0.2045787	3, 282, 784
All Features	*Recommendation*	0.5669792	50, 522, 912
All Audio Features + Lyrics	*RecommendationByLyrics*	0.5404221	42, 753, 312
All Audio Features + Location	*RecommendationByLocation*	0.5387914	41, 540, 384
All Audio Features + Year	*RecommendationByYear*	0.4962294	41, 409, 312
All Audio Features + Album Artwork	*RecommendationByAlbums*	0.4875399	42, 753, 312
All Audio Features	*RecommendationByAudioOnly*	0.4848066	39, 311, 136

Note. These results are obtained by averaging the cosine similarity of every track in the testing split. The total parameters for the recommendation models do not include the parameters of the models they combine.

Table 7

Recommendation examples

Artist	Genres	Origin	Years Active
33Miles	**ccm, christian music, worship**	**Franklin, TN, USA**	**2005-present**
Britt Nicole	ccm, christian music, worship	Kannapolis, NC, USA	2003-present
Plumb	ccm, christian music, worship	Indianapolis, IN, USA	1997-present
Jamie Grace	ccm, christian music, worship	Atlanta, GA, USA	2009-present
Macklemore & Ryan Lewis	**dance pop, pop, pop rap**	**Seattle, WA, USA**	**2008-2017**
Chiddy Bang	indie pop rap, philly rap, pop rap	Philadelphia, PA, USA	2009-present
Lupe Fiasco	chicago rap, pop rap, rap, hip hop	Chicago, IL, USA	2000-present
Mike Posner	dance pop, pop, pop rap	Detroit, MI, USA	2008-present
Three 6 Mafia	**crunk, dirty south rap, gangster rap**	**Memphis, TN, USA**	**1991-2012**
Bun B	deep southern trap, dirty south rap	Port Arthur, TX, USA	1987-present
Mike Jones	dirty south rap, gangster rap, hip hop	Houston, TX, USA	2001-present
Slim Thug	crunk, deep southern trap, dirty south rap	Houston, TX, USA	1998-present
Opeth	**alternative metal, death metal**	**Stockholm, Sweden**	**1989-present**
Storm Corrosion	progressive metal	Stockholm, Sweden	2010-2012
Soen	alternative metal, djent, jazz metal	Sweden	2010-present
Blackfield	neo-progressive, progressive metal	United Kingdom	2001-present
Norwich Cathedral Choir	**british choir, cathedral choir**	**Norwich, England**	**Unknown**
Boston Symphony Orchestra	classical, classical performance, orchestra	Boston, MA, USA	1881-present
St. Paul's Cathedral Choir	british choir, cathedral choir, choral	London, England	1127-present
Choir of King's College, Cambridge	british choir, choral	Cambridge, England	Unknown
Chad Brownlee	**canadian country, contemporary country**	**BC, Canada**	**2003-present**
David Nail	contemporary country, country, country pop	Kennett, MO, USA	2001-present
Jana Kramer	contemporary country, country, country dawn	Rochester Hills, MI, USA	2002-present
High Valley	alberta country, canadian country	Alberta, Canada	1997-present
Aretha Franklin	**classic soul, jazz blues, memphis soul**	**Memphis, TN, USA**	**1956-2018**
Wilson Pickett	classic rock, classic soul, memphis soul	Prattville, AL, USA	1955-2006
Gladys Knight & The Pips	classic soul, mellow gold, motown	Atlanta, GA, USA	1952-1989
Billy Preston	funk, soul	Houston, TX, USA	1956-2005
Ali Zafar	**desi, desi hip hop, filmi, ghazal**	**Lahore, Pakistan**	**2003-present**
Rahat Fateh Ali Khan	desi, filmi, modern bollywood	Faisalabad, Pakistan	1985-present
Atif Aslam	desi, filmi, modern bollywood	Wazirabad, Pakistan	2004-present
Kailash Kher	desi, filmi, indian folk	Meerut, India	2003-present
Angeline Quinto	**classic opm, opm**	**Manila, Philippines**	**2003-present**
Juris	classic opm, opm	Davao City, Philippines	2003-present
Gary Valenciano	classic opm, opm, papuri	Manila, Philippines	1978-present
Toni Gonzaga	classic opm, opm, papuri	Rizal, Philippines	1997-present

Note. The context artist is bolded.

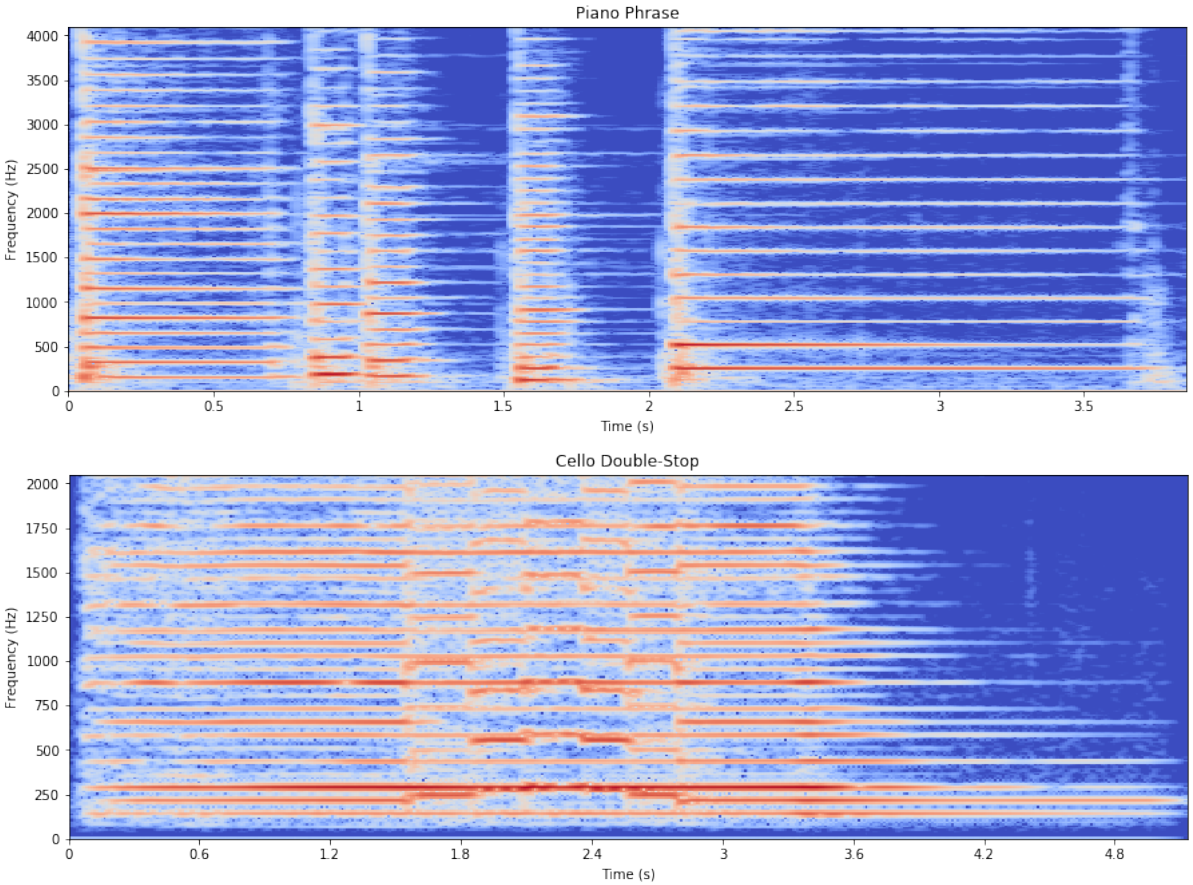

Figure 1. Top: STFT spectrogram of a short piano phrase. The melody can be obtained by extracting the fundamental frequencies of each segment to determine the notes and analyzing the segments to determine the duration. The harmonics of each note are clearly defined as horizontal lines. Bottom: STFT spectrogram of a cello double-stop. The loudest frequency appears to be a harmonic and not a fundamental. The lowest frequency is the persistent D string and the pyramid pattern is made from the climb and fall phrase played on the A string ($A_3 - B_3 - C_4 - B_3 - A_3$).

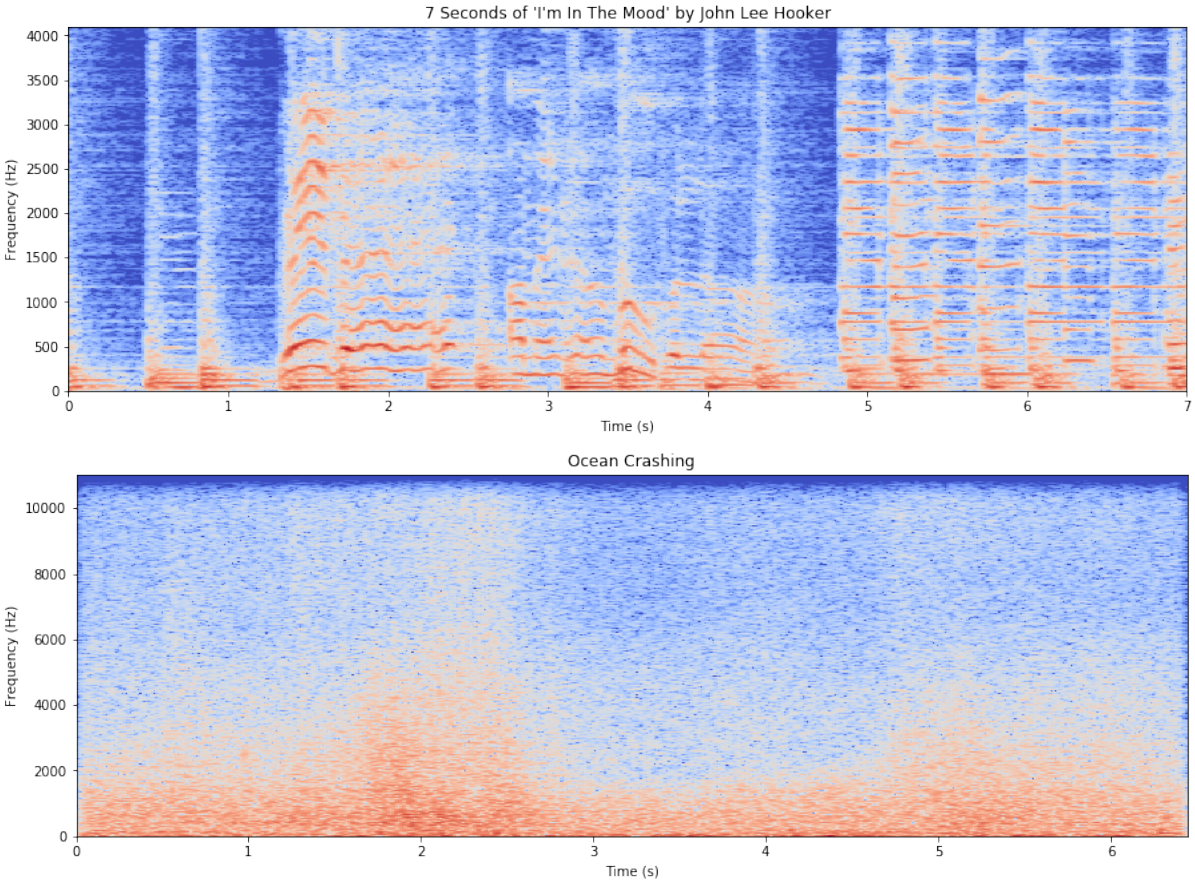

Figure 2. Top: STFT spectrogram of a blues song. The onsets of the drum beats are clear from the vertical bars and the melody is visible horizontally. The rhythm of a song is quite clear; the tempo and time signatures can be estimated from this representation. Note, the harmonics of the singing voice are represented as wavy lines. Bottom: STFT spectrogram of the ocean. Stochastic sounds do not have clearly defined structure. If the phase spectrogram was included, it would very much look like random noise.

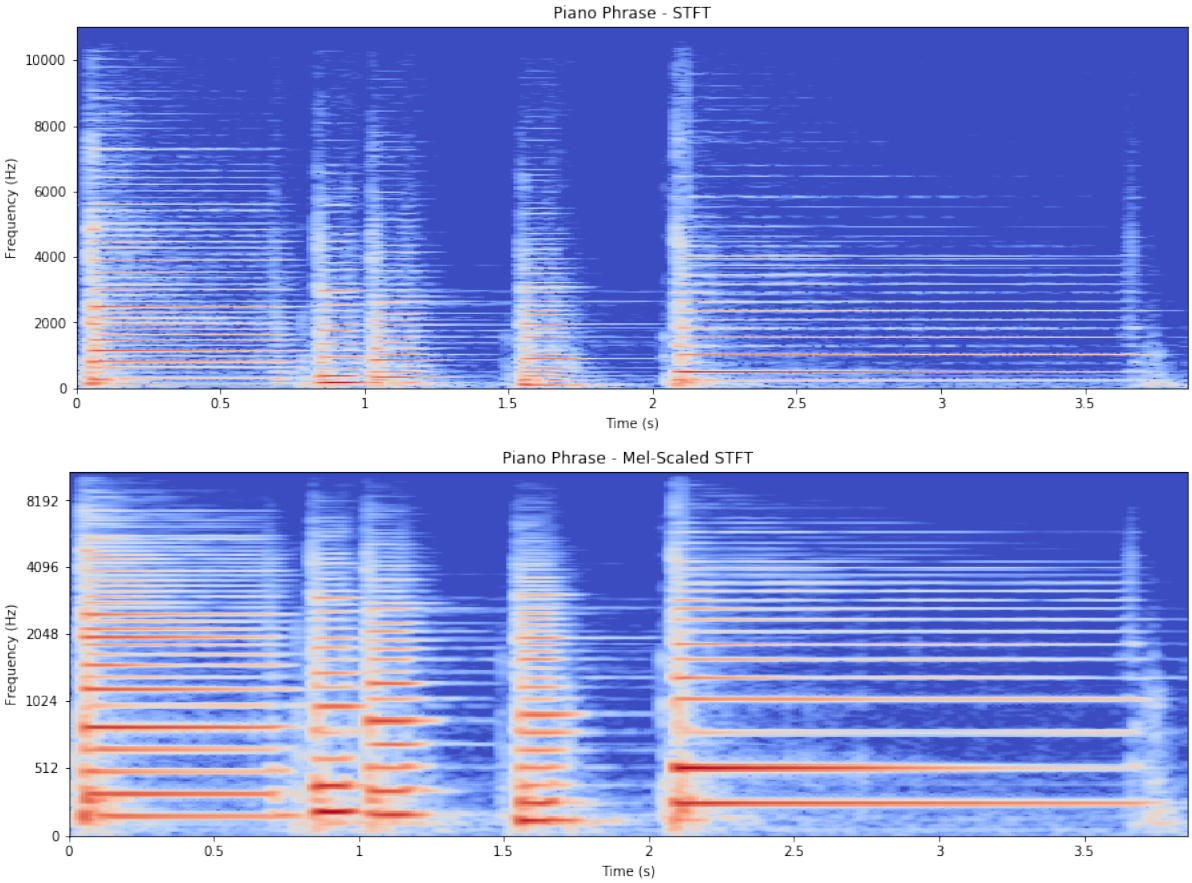

Figure 3. Top: The STFT spectrogram of a short piano phrase. Bottom: The same STFT spectrogram mapped onto the mel basis. Note that the frequency axis scales logarithmically. The lower frequencies are accentuated.

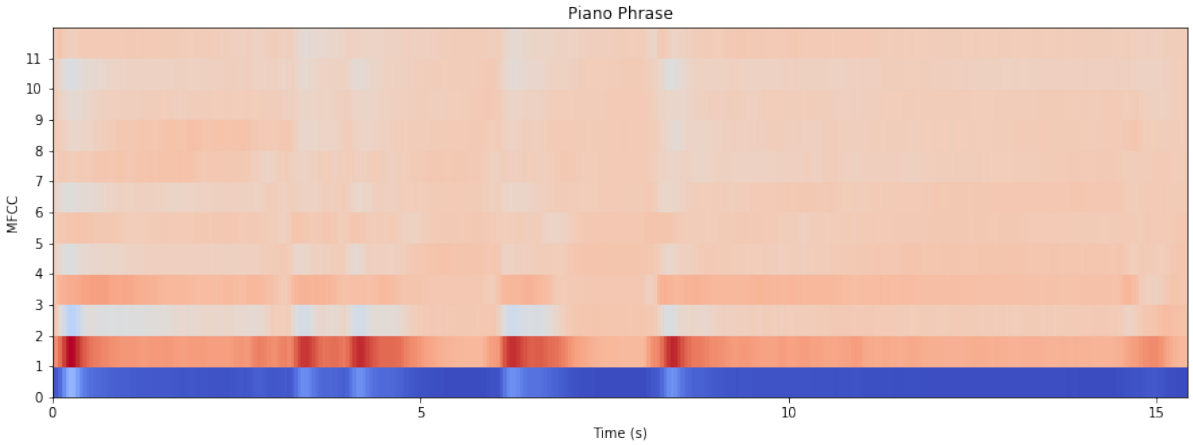

Figure 4. The first 12 MFCCs of a short piano phrase.

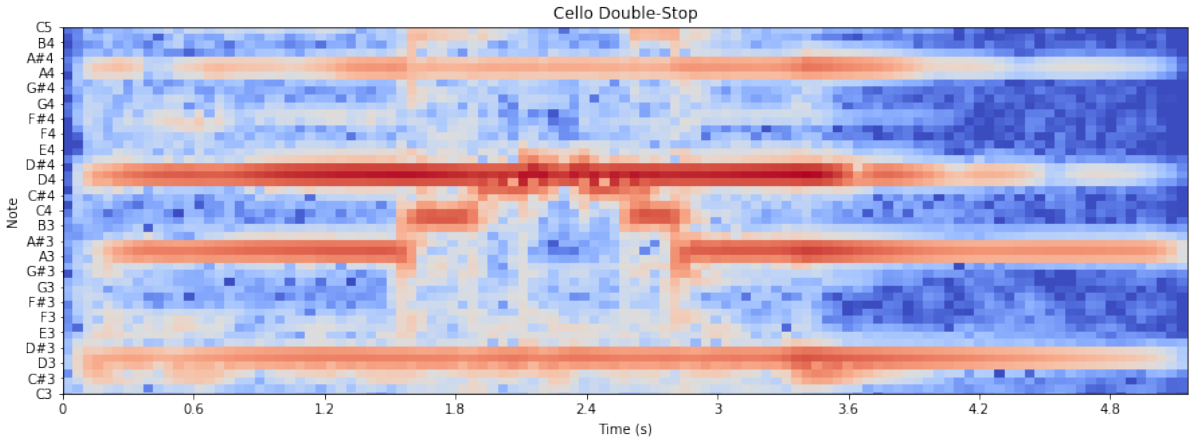

Figure 5. The cello double-stop notes consist of a D_3 played below a seven note phrase that goes $A_3 - B_3 - C_4 - D_4 - C_4 - B_3 - A_3$. These notes are clearly labeled in the spectrogram.

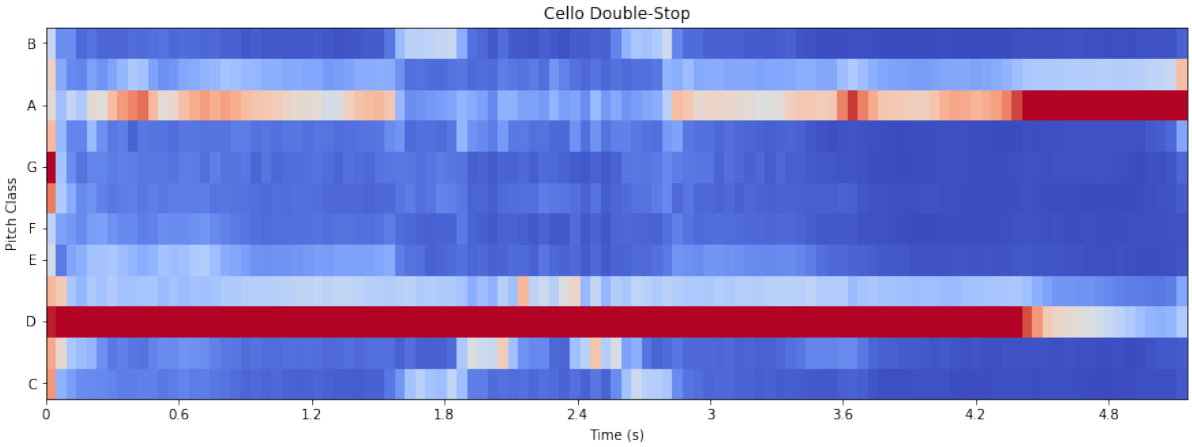

Figure 6. Chromagram of a cello double-stop. The highest pitch class energies belong to D and A.

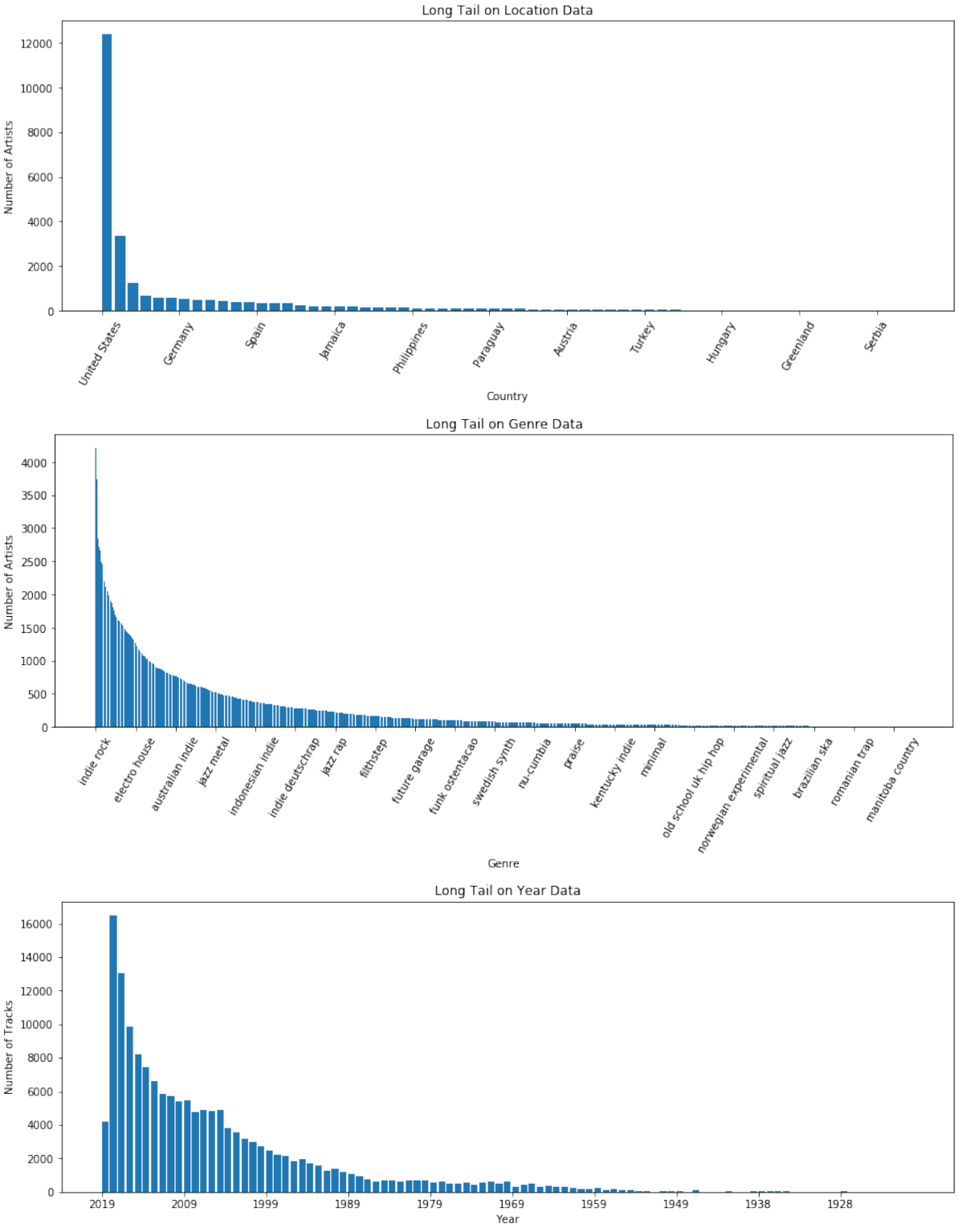

Figure 7. Long tail visualized on location, genre, and year data of the *WebAPI* dataset.

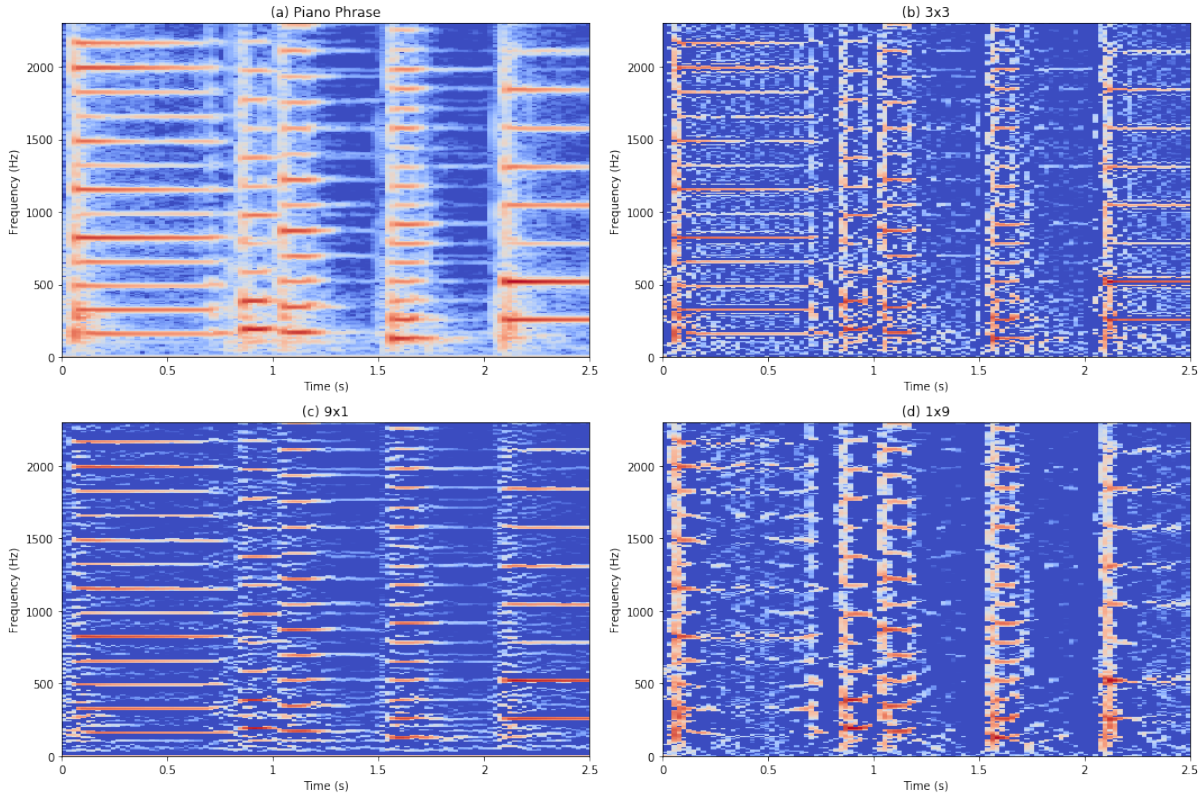

Figure 8. Edge detecting kernels of different sizes applied to the short piano phrase shown in (a). Each convolution is followed by a ReLU activation function which filters out much of the noise. All kernels have a weight of 8 in the middle pixel and -1 elsewhere. The 3×3 kernel of (b) detects and amplifies both horizontal edges (frequencies, harmonies) and vertical edges (attacks, onsets). However, the frequencies in (c) and the onsets in (d) are much more clear then they are in (b).

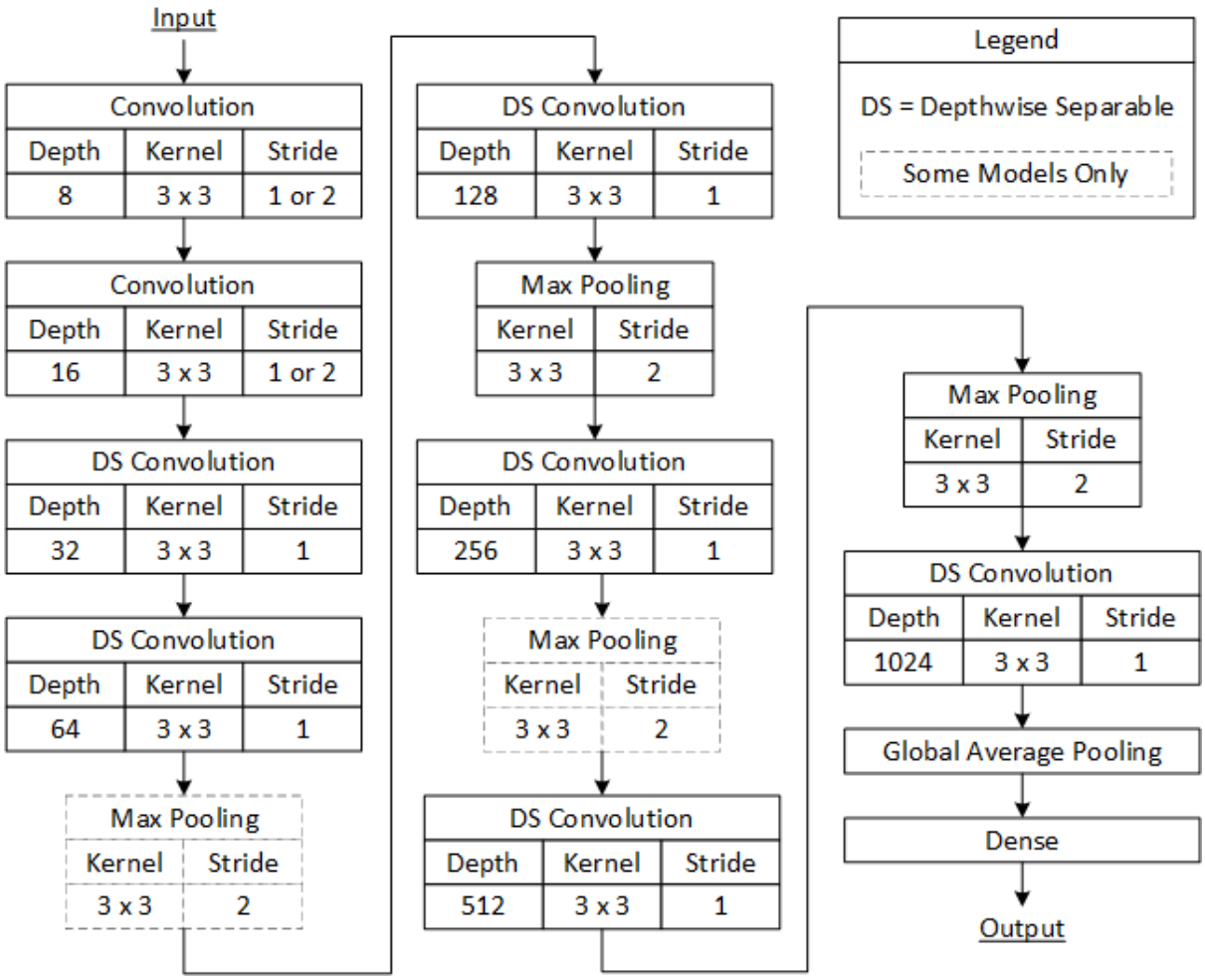

Figure 9. The *Simple* architecture. A model with this architecture trained on the *Cifar-100* dataset does not contain the first two max pooling layers because the images are small (32×32) and do not need to be reduced as much as spectrograms. The output size and activation function of this model depend on the dataset and task. Not shown: ReLU activation functions and batch normalization.

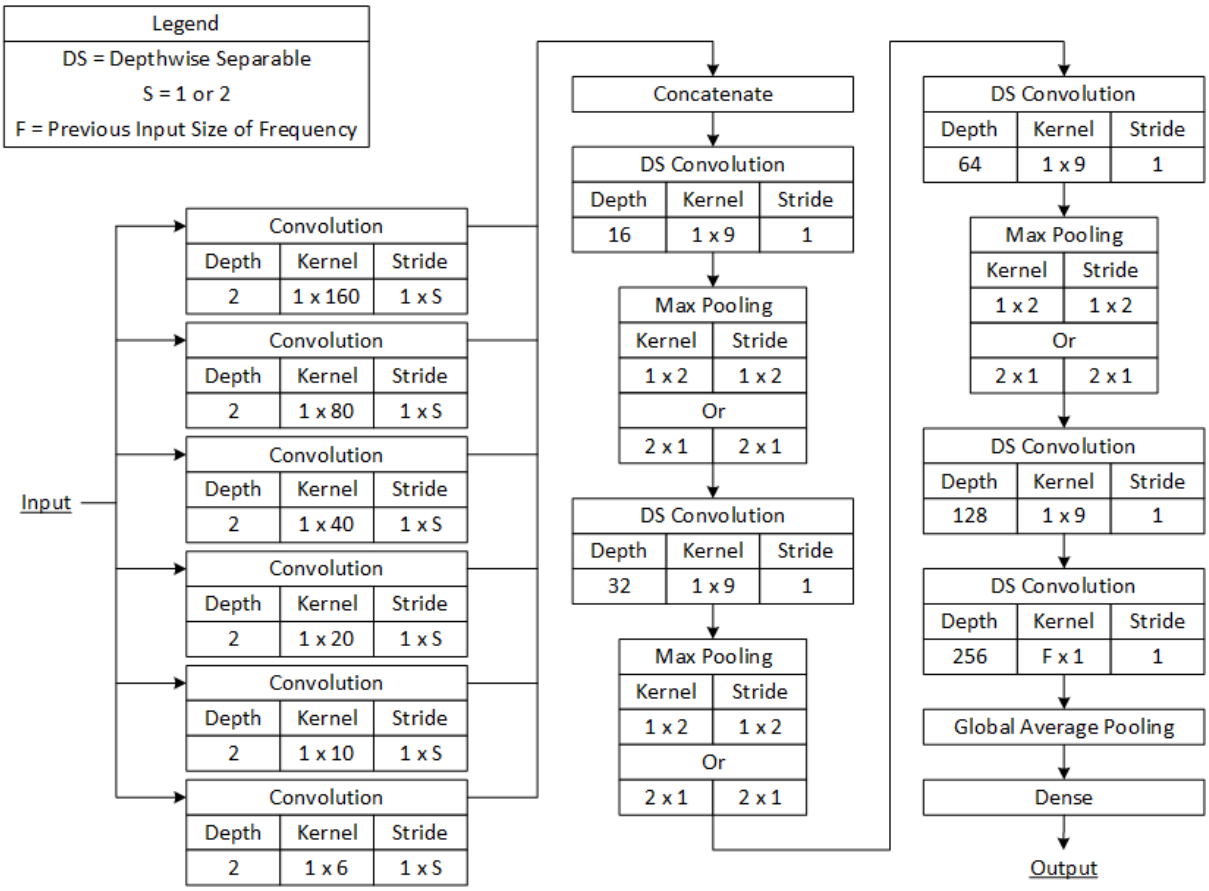

Figure 10. The *Time* architecture. A model with this architecture trained on the *Cifar-100* dataset uses strides of 1×1 on the six input convolutions. All models that use spectrograms as input use strides of 1×2 on the six input convolutions to reduce the time dimension. Also, the max pooling layers reduce the frequency dimension for all inputs except for chromagrams and MFCCs which are already reduced in frequency. The output size and activation function of this model depends on the dataset and task. Not shown: ReLU activation functions and batch normalization.

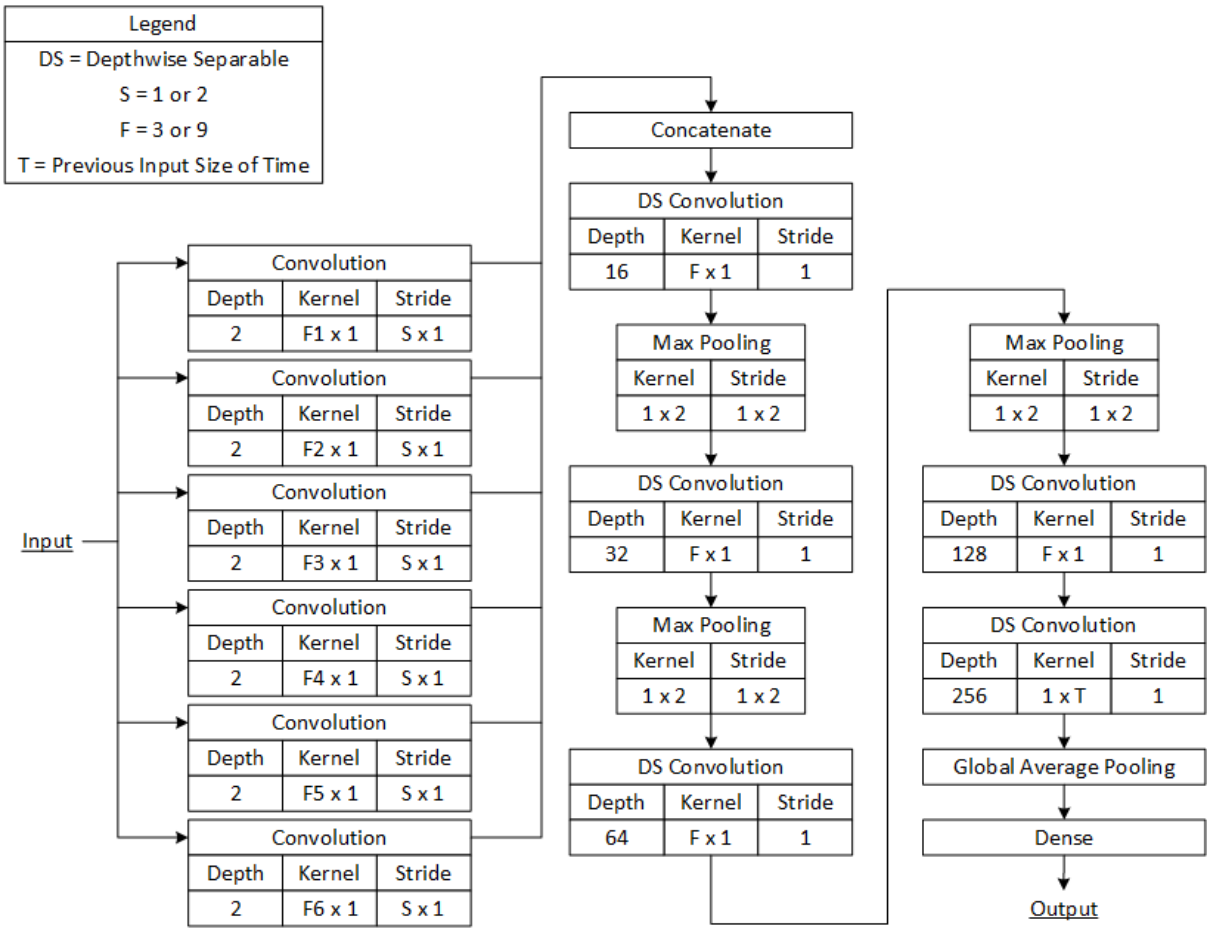

Figure 11. The *Freq* architecture. A model with this architecture trained on the *Cifar-100* dataset or with chromagrams or MFCCs as input use strides of 1×1 on the six input convolutions. Otherwise, strides of 2×1 are used to reduce the frequency dimension. Unlike the *Time* architecture which uses the same kernel shape regardless of the input, the kernels of the *Freq* architecture depend on the input. This is because all spectrograms have a time dimension of 643 from a constant track length of 30 seconds, but the frequency dimension varies depending on the spectral representation. The output size and activation function of this model depends on the dataset and task. Not shown: ReLU activation functions and batch normalization.

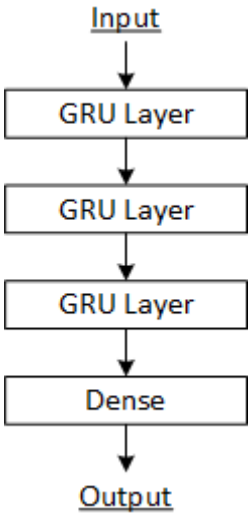

Figure 12. The *RNN* architecture. GRU layers always have a dimension of 256. The output size and activation function of this model depends on the dataset and task.

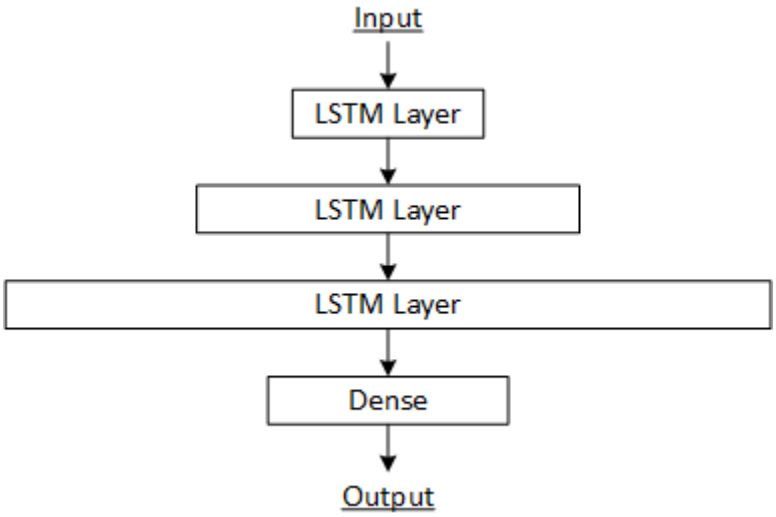

Figure 13. The *LargeRNN* architecture. LSTM layers increase in size. The output size and activation function of this model depends on the dataset and task.

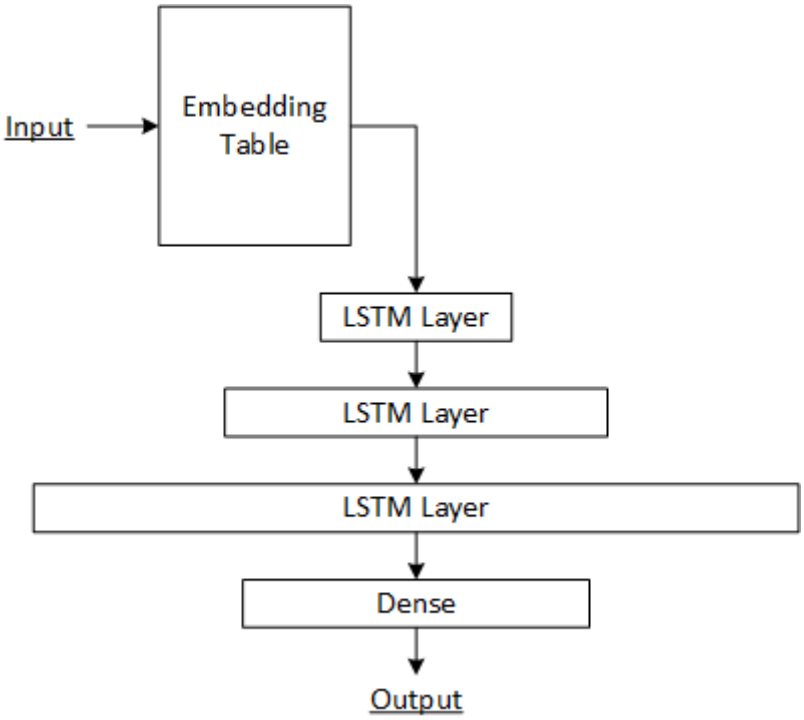

Figure 14. The *Lyrics* architecture. This architecture is the same as the *LargeRNN* architecture but with an embedding layer at the input for each word in the vocabulary. The final dense layer size matches the chosen artist embedding vector dimension (800) and has no activation at the output.

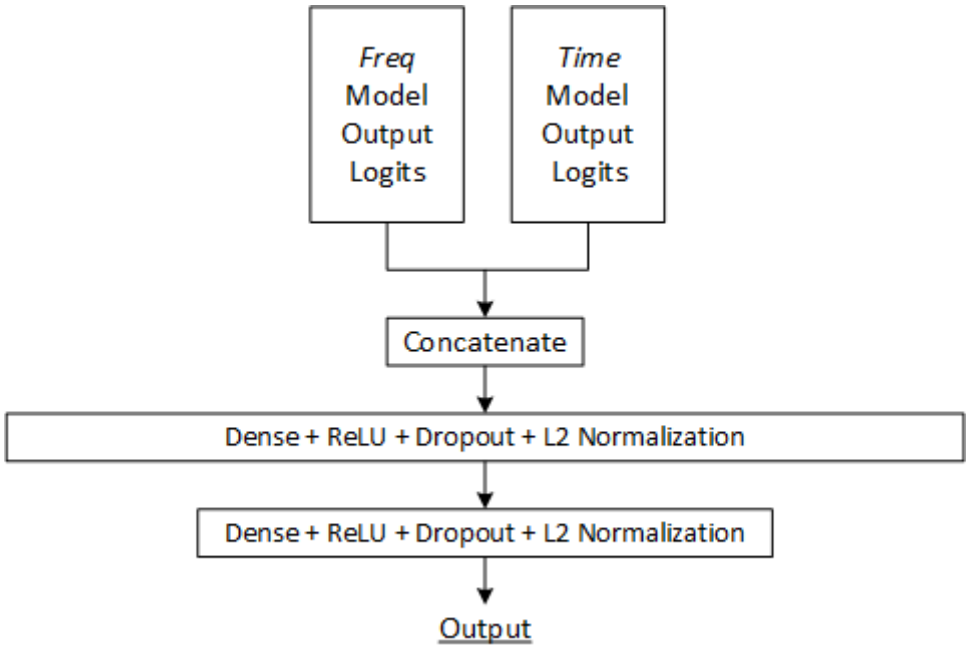

Figure 15. The *TimeFreq* architecture.

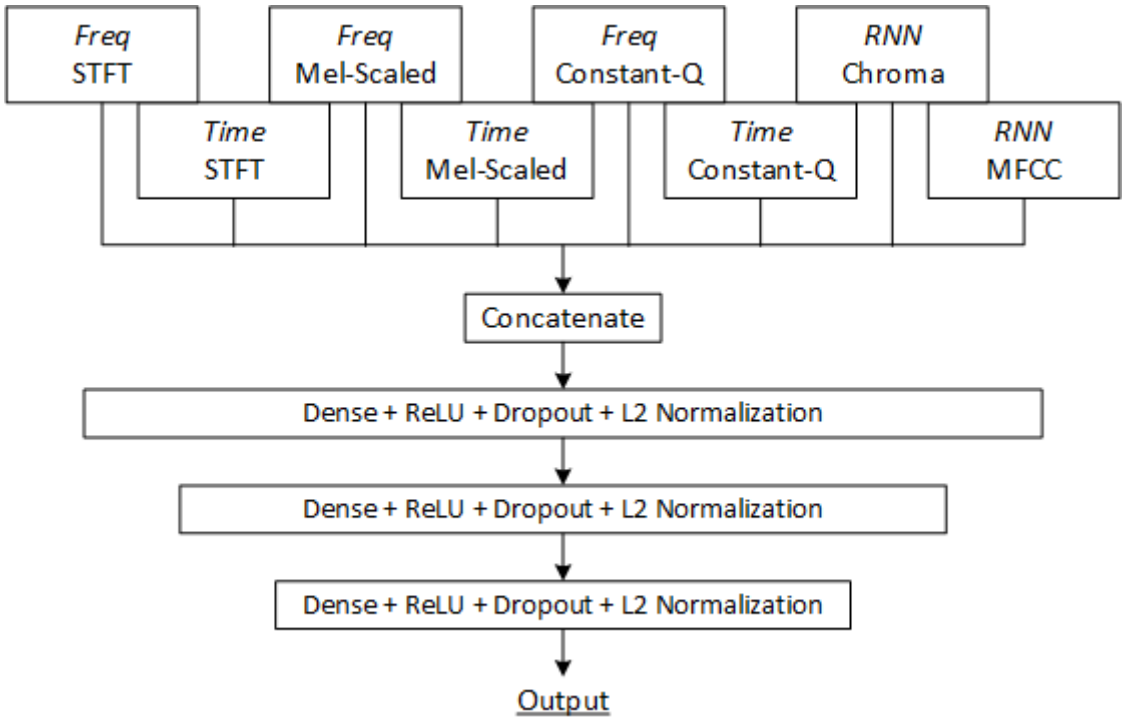

Figure 16. The *GenreEnsemble* architecture.

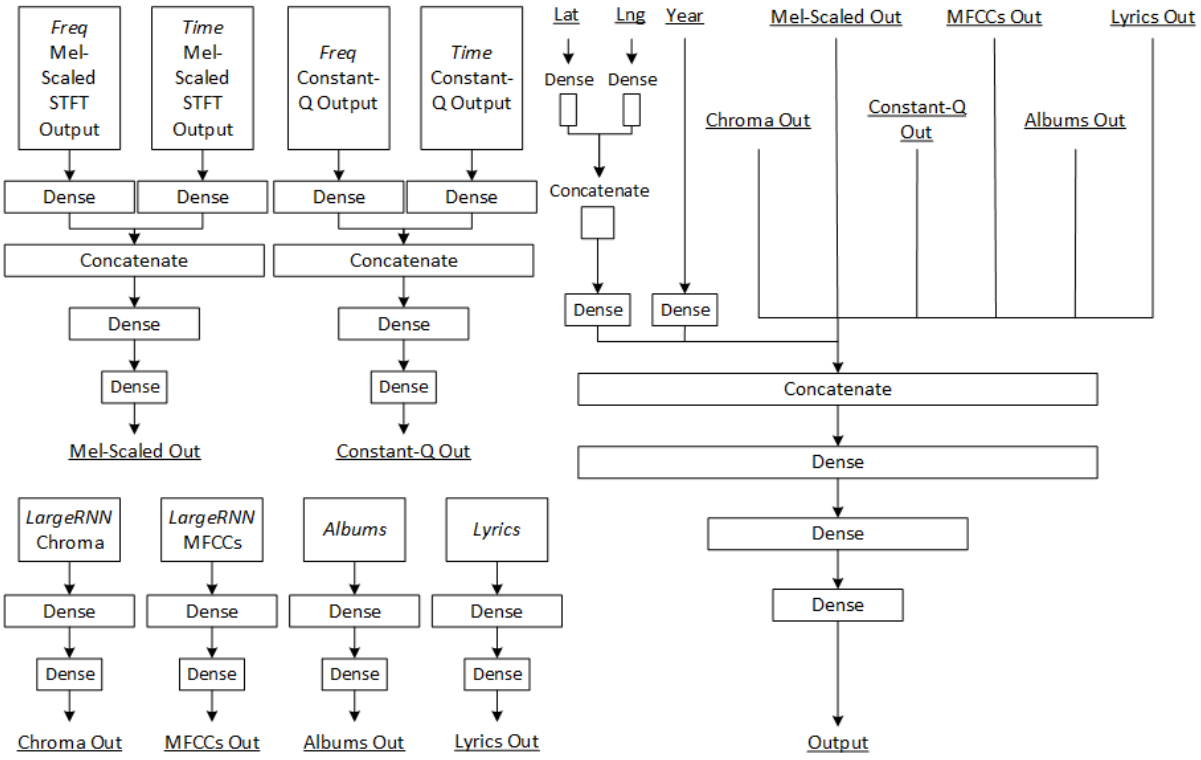

Figure 17. The *Recommendation* architecture. The block widths roughly correspond to the size of each layer. Every dense layer is followed by a ReLU activation function. The inputs from the *Time*, *Freq*, *LargeRNN*, *Albums*, and *Lyrics* models are 800 dimensional vectors which are reduced by deep networks into 512 dimensional vectors. Latitude (lat) and longitude (lng) inputs are combined and increased to 512 dimensional vectors by dense layers. The year input is simply increased by supplying it as input into a 512 dimensional dense layer. All eight 512 dimensional vectors are concatenated before being fed into the final three dense layers.

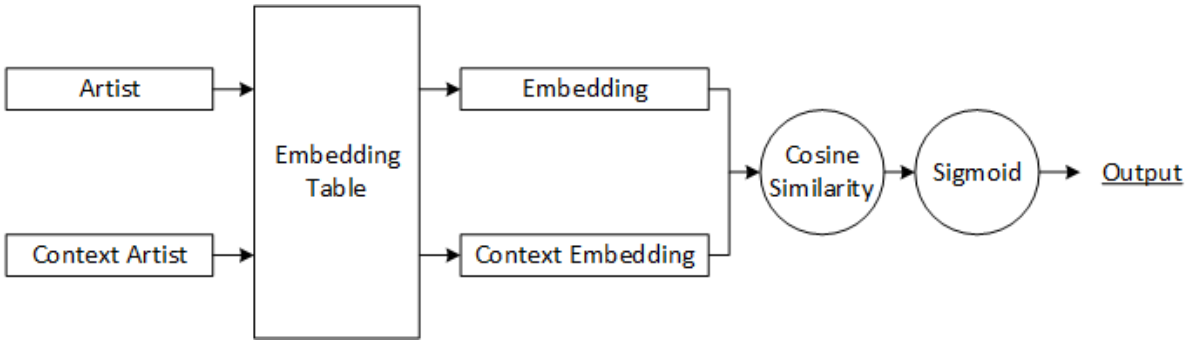

Figure 18. The artist embedding model. The context artist is either a related artist or a random artist. The embedding table is trained so that artist embeddings of related artists are similar, pushing the output to 1, and artist embeddings of random artist are dissimilar, pushing the output to 0.

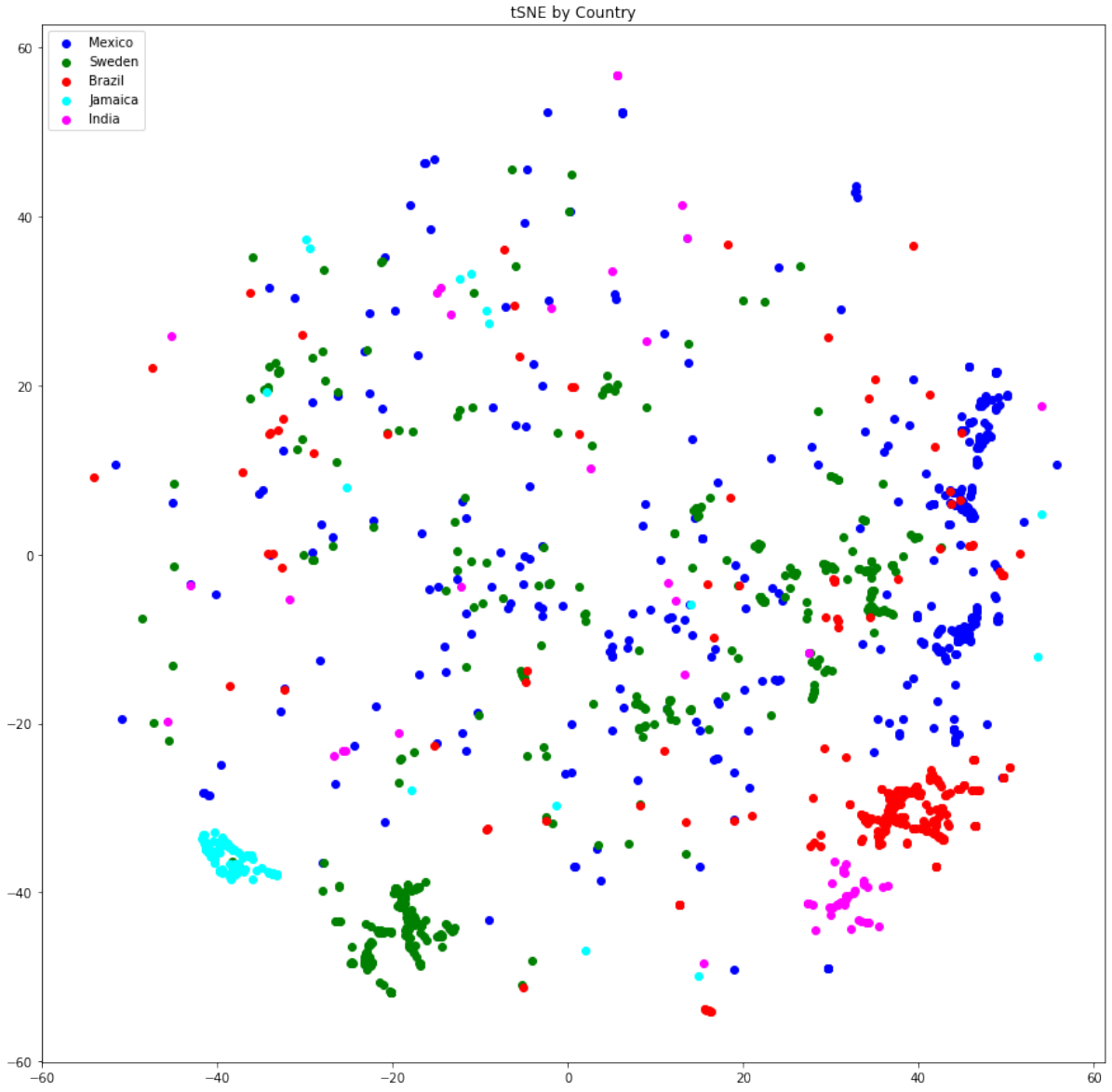

Figure 19. t-SNE by Country

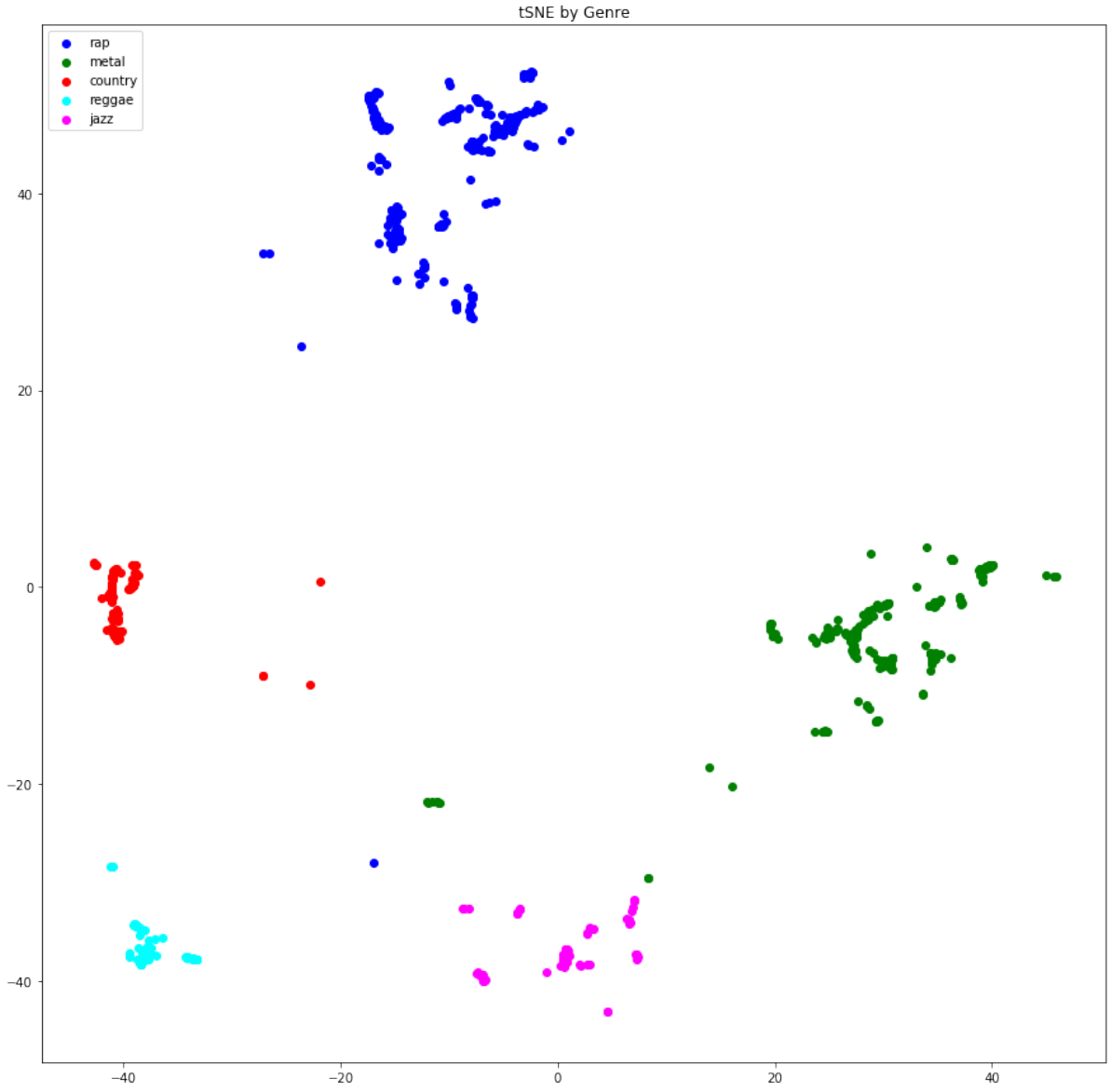

Figure 20. t-SNE by Genre

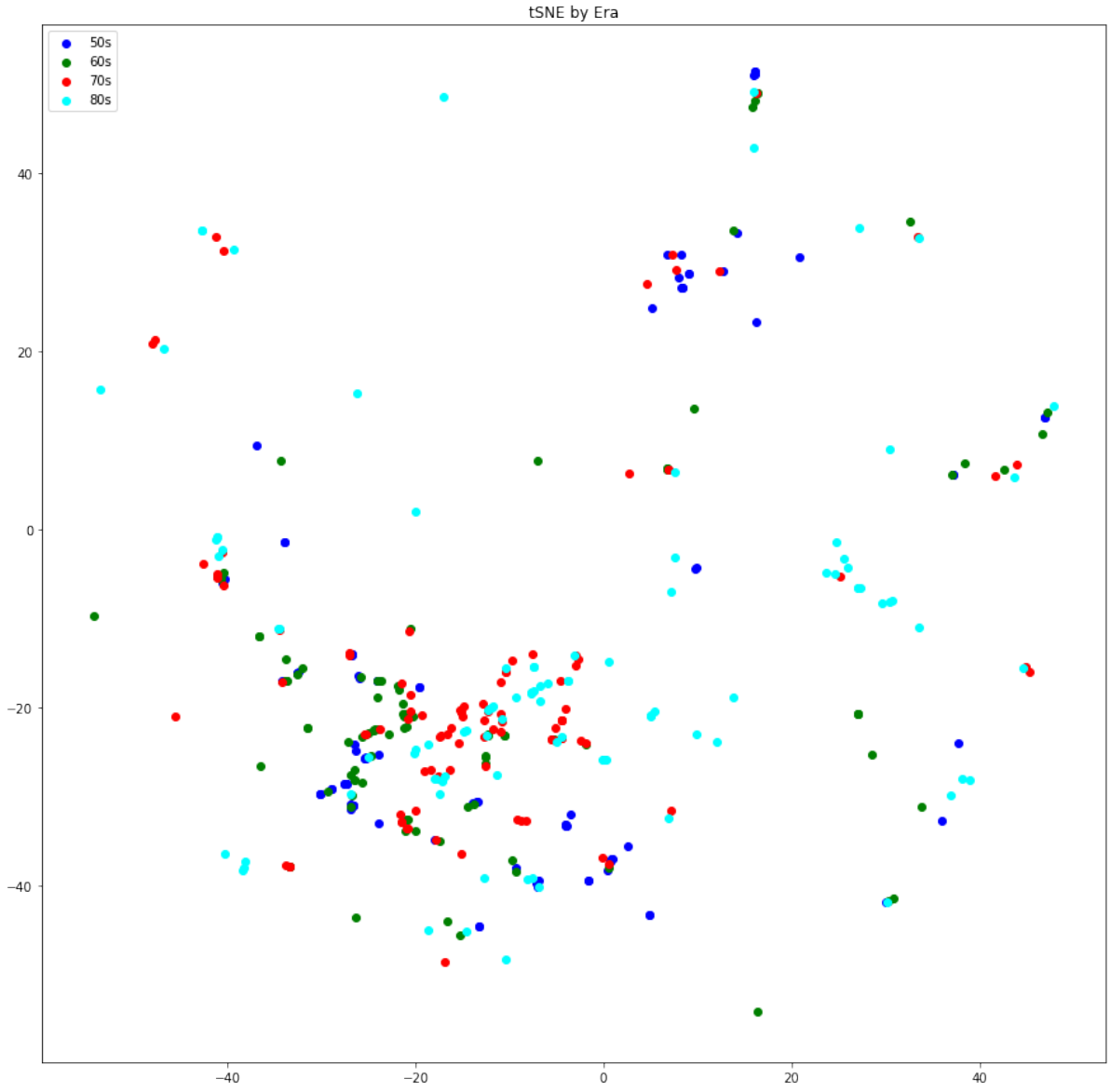

Figure 21. t-SNE by Era

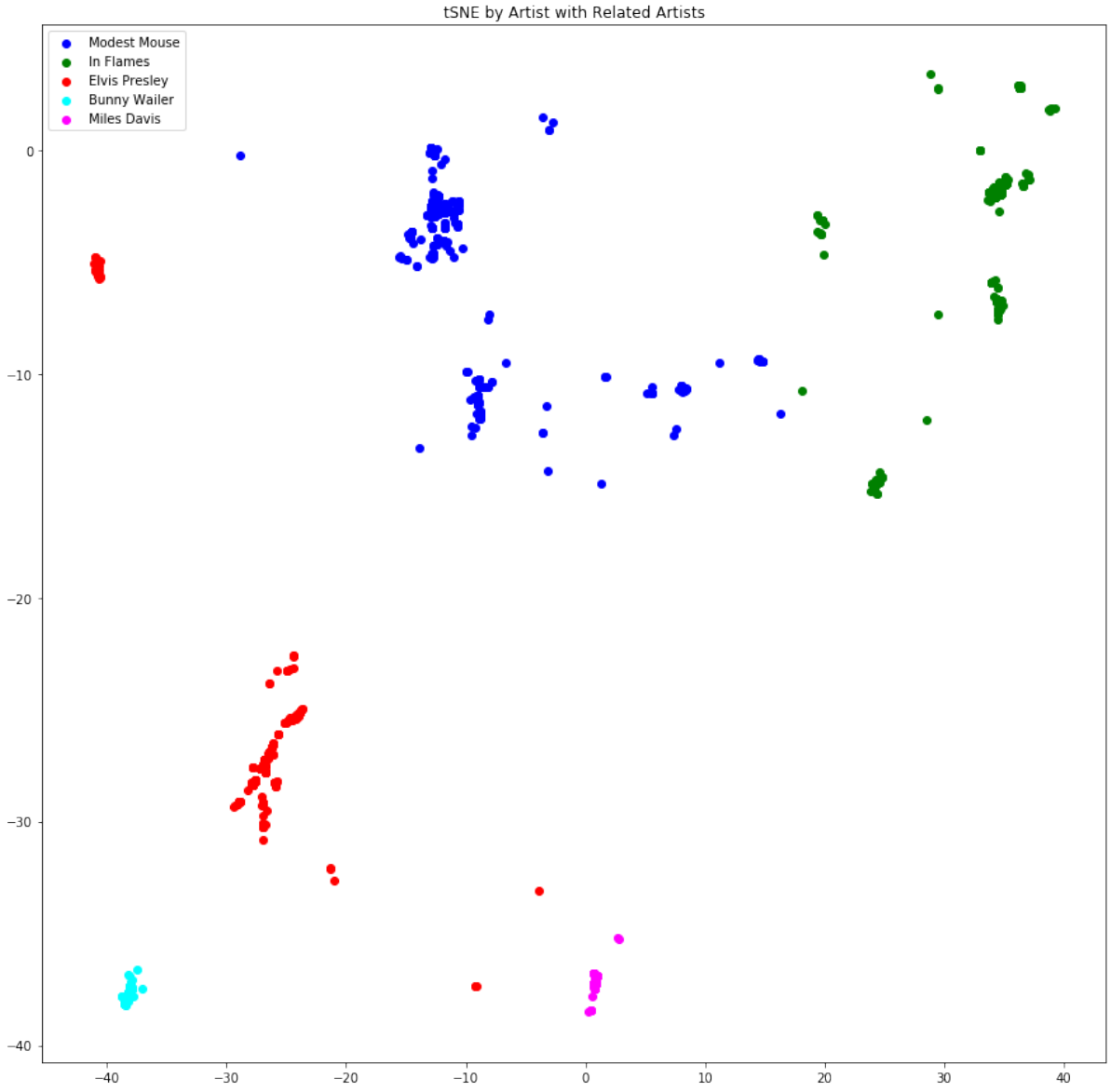

Figure 22. t-SNE by Artist