Predict Song Popularity

Myrto-Aglaia Tsiamasioti

Academic Supervisor:
Dr. Prodromos Malakasiotis

Company Supervisor:
Pantelis Vikatos

School of Information Sciences and Technology
Department of Informatics

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science
in

Data Science

Athens, November 2022

Abstract

With the rise of music streaming platforms in recent years, quantifying and digitalizing a song's defining parameters to extract information on its potential success is a very beneficial practice when deciding how much budget to allocate for its promotion or discovering promising artists. The objective of this thesis is to infer the potential success of a given song, by training a model for the estimation of its popularity using low and high level audio features. In total, four different types of data are provided, concerning songs released from the start of 2017 to the end of 2018. Spotify Tags, Genres, the Mel Spectrogram and monthly YouTube Views of a track are all utilized for predicting a song's popularity both before and after its release. This thesis aims to utilize the provided data in both machine and deep learning scenarios. For that purpose, different types of models were developed for the two approaches respectively. Machine Learning Tree-Based models like Random Forest, LightGBM and XGBoost were implemented and trained on a tabular format dataset. Moreover, deep neural networks such as Convolutional and Recurrent Neural Networks and hybrid combinations of them were developed to be trained both on image and sequence data. Through the evaluation of their performance, it was found that regardless of their much different structure and architecture, both of the approaches have comparable performance. Even though adequate results were achieved, the detection of hit songs specifically was and still remains a much bigger challenge, due to their under-representation not only in the provided dataset but also in real life.

Περίληψη

Με την άνοδο των πλατφορμών συνεχούς ροής μουσικής τα τελευταία χρόνια, η ποσοτικοποίηση και η ψηφιοποίηση των καθοριστικών παραμέτρων ενός τραγουδιού για την εξαγωγή πληροφοριών σχετικά με την πιθανή επιτυχία του είναι μια πολύ ωφέλιμη πρακτική για τη λήψη αποφάσεων σχετικά με τον προϋπολογισμό που πρέπει να διατεθεί για την προώθησή του ή την ανακάλυψη υποσχόμενων καλλιτεχνών. Η παρούσα διπλωματική εργασία επικεντρώνεται στην εξαγωγή συμπερασμάτων σχετικά με τη δυνητική επιτυχία ενός συγκεκριμένου τραγουδιού, μέσω της εκπαίδευσης ενός μοντέλου για την εκτίμηση της δημοτικότητάς του, χρησιμοποιώντας χαρακτηριστικά ήχου χαμηλού και υψηλού επιπέδου. Συνολικά, παρέχονται τέσσερις διαφορετικοί τύποι δεδομένων, που αφορούν τραγούδια που κυκλοφόρησαν από τις αρχές του 2017 έως το τέλος του 2018. Τα Spotify Tags, τα Genres, το Mel Spectrogram και οι μηνιαίες προβολές ενός κομματιού στο YouTube χρησιμοποιούνται για την πρόβλεψη της δημοτικότητας ενός τραγουδιού τόσο πριν όσο και μετά την κυκλοφορία του. Η παρούσα διατριβή αποσκοπεί στην αξιοποίηση των παρεχόμενων δεδομένων τόσο σε σενάρια μηχανικής όσο και σε σενάρια βαθιάς μάθησης. Για το σκοπό αυτό, αναπτύχθηκαν διαφορετικοί τύποι μοντέλων για τις δύο προσεγγίσεις αντίστοιχα. Εφαρμόστηκαν και εκπαιδεύτηκαν tree-based μοντέλα μηχανικής μάθησης, όπως τα Random Forest, LightGBM και XGBoost, στο σύνολο των δεδομένων σε μορφή πίνακα, καθώς και βαθιά νευρωνικά δίκτυα, όπως τα Convolutional και Recurrent Neural Networks και υβριδικοί συνδυασμοί τους, που δημιουργήθηκαν για να εκπαιδευτούν τόσο σε δεδομένα εικόνας όσο και σε δεδομένα ακολουθίας. Μέσω της αξιολόγησης της επίδοσης τους, διαπιστώθηκε ότι ανεξάρτητα από την πολύ διαφορετική δομή και αρχιτεκτονική τους, και οι δύο προσεγγίσεις έχουν παρόμοια επίδοση και αποτελέσματα. Παρόλο που επιτεύχθηκαν επαρκή αποτελέσματα, η ανίχνευση των πραγματικά πετυχημένων τραγουδιών ήταν και παραμένει μια πολύ μεγαλύτερη πρόκληση, λόγω της υποεκπροσώπησής τους όχι μόνο στο σύνολο δεδομένων που δόθηκε αλλά και στην πραγματική ζωή.

Acknowledgements

I would like to thank the following people, whose assistance and support was invaluable to the completion of the following thesis.

First of all, I would like to express my gratitude for my academic supervisor's Dr. Prodromos Malakasiotis's guidance, whose insight and expertise were crucial in overcoming the various challenges encountered in this endeavor. Moreover, I would like to thank Mr. Pantelis Vikatos of Orfium, for introducing me to the thesis theme and giving me the opportunity to explore and work with data formats which are commonly encountered in the music industry. Last but definitely not least, I would like to thank my family and friends for their continuous support and encouragement throughout the writing of this thesis.

Contents

Chapter 1

Introduction

Music is a huge component in our everyday lives. It is present in television, social media applications, and is a great facilitator bringing cultures and people closer together. From vinyl records, cassette tapes and CD disks, music was distributed in multiple ways to all fans of tunes. However, as technology advances, the new way of listening to music is online, with applications such as Spotify, YouTube and Apple Music being on the rise. Spotify is the world's biggest music streaming platform, with over 80 million tracks and 433 million active users worldwide. As of October 2022, the Business of Apps website reports that Spotify only has increased its annual revenue by 22% in 2021 to €9.66 billion, tripling its revenue in the past five years.[1] The exponential growth of these streaming applications has made music more accessible than ever, only with a simple subscription. Listeners are granted access to a huge library of songs, listening to more music than ever before.

With music being accessible to everyone, the information concerning its consumption is gathered in large, complex datasets. Essentially, this interplay between music and data leads to the new era of the digitization of music, changing many aspects of the production and marketing of music. A song can be quantified, cut down to its parts, and the information extracted from these parts can be used for judging its prospect of success, consequently "ignoring" the artistry that goes behind it. Indeed, the better businesses like Spotify manage to tune their algorithms by quantifying music, the better they can adjust their business model accordingly in order to be more a profitable corporation. With the promise of profit, many have jumped to the opportunity of creating new methods of producing and marketing music, judging by how much revenue a song can generate. This can help us decide how much budget a new song may need for promotion purposes.

1.1 Song Popularity

The popularity of a song cannot be defined by a single variable. In fact, it varies to alternative targets such as high ranks in charts like billboard, or the number of Spotify monthly

[1] Source: https://www.businessofapps.com/data/Spotify-statistics/

listeners. Spotify has a unique popularity feature for quantifying the song's success by taking into account a multiple of both internal and external factors. As stated in the Spotify API [2],

> *"The popularity of a track is a value between 0 and 100, with 100 being the most popular. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are."*

According to the API, songs that are being played a lot in the present will have a higher popularity than songs that were played a lot in the past. The popularity value may lag actual popularity by a few days: the value is not updated in real time, but is constantly renewed. Defining a song's popularity can be critical for many reasons. It is one of the most important factors indicating how successful a song will be. In addition, popularity allows the song industry to find potentially successful songs, promising songwriters and composers, to allocate budget for promotion, and to identify key elements that are pivotal for the success of a song. The emerging field that aims at predicting the success of songs before they are released on the market is called Hit Song Science.

1.2 Problem Statement

In order to extract information about a song, we need to "cut it down" to its smaller parts. A song can have many features, usually divided into physical and perceptual features (Table 1.1). Physical features, also known as low-level features, are obtained from the audio content, computed directly from the raw waveform of the song. They are considered to be at the lowest level of abstraction, as extracting useful information from such features is not easy. On the other hand, perceptual or high-level features are symbolic terms based on perception of sound by humans, and as a result are much better comprehended. In addition to these features, mid-level features are also sometimes extracted; they are called mid-level as they can occur as waveform representations of sound, but also contain symbolic representations. The goal of this thesis is to predict hit songs based on various audio features, as well as taking into consideration the song's YouTube Views thus far. Additionally, a closer look is taken into how much a song's attributes can contribute to a track's popularity.

Level of Abstraction	**Instances of Audio Features**
High	loudness, instrumentalness, key, tempo
Mid	pitch, beat-related descriptors, note onsets, fluctuation patterns
Low	Mel-Spectrograms, Mel-Frequency Cepstral Coefficients

TABLE 1.1: Types of Audio Features

[2]Spotify API: https://hexdocs.pm/Spotify_web_api/Spotify.Tracks.html

1.3 Thesis Outline

For the aims of this thesis, high-level and low-level features will be used to describe a song. A combination of different types of data concerning a total of 10753 songs are provided, that were released from the start of 2017 to the end of 2018. These include features in the form of Spotify tags (high-level features), the song's genres, and the Mel-Spectrogram (low-level feature) of every song. Additionally, monthly YouTube views for each song are given, from the start of 2019 until May of 2022. Mel-frequency cepstral coefficients (MFCCs) are also used as features, extracted from the Mel-Spectrograms.

Relevant background work on popularity prediction is covered in Chapter 2. This work covers experiments made using just high-level features specifically for popularity prediction, as well as various projects implementing low-level ones with different objectives.

In Chapter 3, the provided data are explained in further detail and exploratory analysis is performed in order for the reader to better comprehend the various different types of data used for prediction.

The methods and models used for the prediction of song popularity are analysed in detail in Chapter 4.

Then, the results of the conducted experiments are presented in Chapter 5, where the quality of the predictions as well as the importance of the most influential features in prediction are discussed.

Finally, a conclusion is reached in Chapter 6 concerning how the models performed and how the work of this thesis could be expanded in the future.

Chapter 2

Related Work

There is a wide range of song prediction tasks, as predicting song popularity is barely a new practice. Inspiration was taken from many examples of projects, some using just low frequency data such as Spotify Tags, and others being more complex by implementing Neural Networks on Mel-Spectrograms. The ones that had the biggest contribution to the current thesis are mentioned below.

2.1 High-Level Audio Features

The underlying assumption in Hit Song Science is that hit songs are similar with respect to their features. That would imply that there are some features, with high predictive power when it comes to predicting a song's popularity. There is a lot of literature tackling this, asking questions such as: are there certain characteristics for hit songs? What characteristic has the largest influence on the song's success? The most relevant models in popularity prediction using just audio features and their associated papers are discussed here. Most projects are classification tasks, predicting whether a song is a hit or not. Inspiration was taken from how these projects approach popularity prediction, and these ideas were modified to fit our regression task.

Nasreldin et al. [1] aimed to predict whether a song will be included in the Billboard Hot 100 chart or not, using machine learning methods. A subset of 10,000 from the Million Songs Dataset[1] provided by Columbia University was used, with features relating to audio analysis (e.g. tempo, loudness), the artist (e.g. artist popularity) and the song (e.g. release year). Models like Random Forest [2] and K-Nearest Neighbors [3] were used. The most accurate model was XGBoost [4], which predicted popular songs with 68% accuracy. The authors note that using additional features such as artist location or release date, as well as a larger dataset, could possibly increase the performance of the models.

Similarly, Middlebrook and Sheik [5] compared four models including a simple Multi-Layer Perceptron (MLP) Neural Network using just Spotify Features for training. This project utilized Spotify's API to gather metadata on tracks, as well as the Billboard API to gather information on whether a song made it on the Billboard Hot 100 chart. The models were trained on the combined dataset containing approximately 24.000 songs. Of

[1] Dataset from: http://millionsongdataset.com/

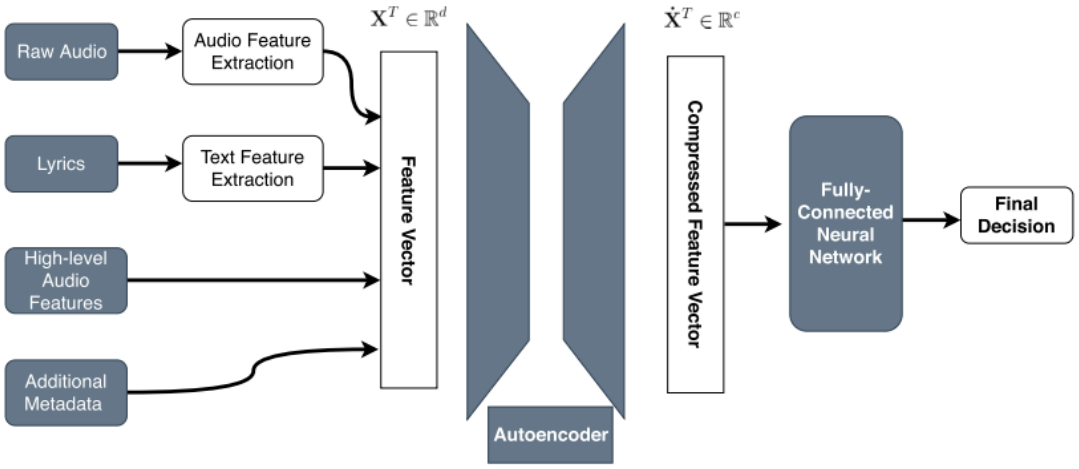
all models used, the Random Forest model ended up outperforming the rest, predicting Billboard song success with 88% accuracy.

2.2 Low-Level Audio Features

A Mel-Spectrogram is an alternative visualization of the waveform of a song. It basically shows the frequencies that make up a sound, from low to high, and how they change over time, from left to right, with all frequencies converted to the Mel scale [6]. Mel-Spectrograms are the main format for audio in prediction tasks. Based on prior research, higher performance is documented using this representation of audio in comparison to others. Additionally, the way this input is constructed makes it easier to use future test data on the models created, as getting the Mel-Spectrograms of a song is standard practice and can be applied on different kinds of audio.

Zangerle et al. [7] added low-level features in training and attempted to see which features are the most useful in popularity song prediction, by experimenting with various combinations of them. The Million Song Dataset was used here as well. They proposed the combination of both low and high-level audio features of songs in a deep neural network. The aim of the experiments was rank prediction, taking into account the rank position of each song in the Billboard Hot 100. Their final deep learning model, combining deep and wide input layers, achieved 75.0% accuracy and utilized features such as Mel-Frequency Cepstral Coefficients, the year of the song release, the artist's voice (if any - the song might be instrumental), its mood and genre. These experiments proposed distinguishing between low- and high-level features before combining them and training them seperately, in order to account for each feature's particularities.

Martín-Gutiérrez et al. [8] proposed using a combined vector of features concerning each song. The features include raw audio signal, represented by Mel-Spectrograms in addition to more representations, the full lyrics of each song, high-level audio features obtained from Spotify and lastly an additional set of meta-data based on the social information of the track. This feature vector was then passed through a compression stage using an Autoencoder (Figure 2.1). The compressed featured vector was used as input to a fully-connected Neural Network. Several experiments from both classification and regression perspectives were conducted in order to determine the success of the song, achieving almost 0.01 Mean Squared Error (MSE) and 83% accuracy respectively on the validation set. The paper suggests that having independent features, which are then later concatenated into a neural network can produce great results. Indeed, Autoencoders are a great way of compressing information from different sources before introducing them to the network and could potentially be a great solution to the current thesis, however they need to be trained on a sufficient amount of data in order to generate useful results. Martín-Gutiérrez et al. trained the model on a total of 101.939 tracks.



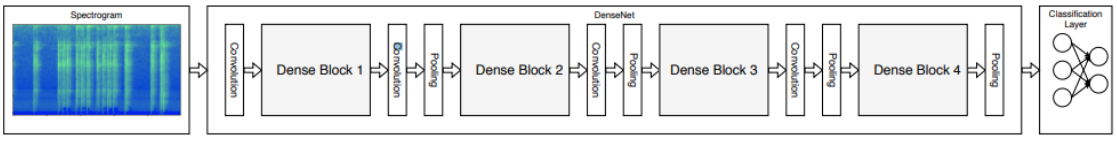
(Image Source)

FIGURE 2.1: Feature compression using an Autoencoder.

Low level features like Mel-Spectrograms are typically utilized in tasks like genre classification, speech emotion recognition, and audio tagging. Although such tasks fall out of the scope of this thesis, they provide useful insights and the most important approaches are briefly described below.

2.2.1 Mel-Spectrograms

Palanisamy, Singhania, and Yao [9] tested different pre-trained standard deep Convolutional Neural Network (CNN) models on three different Audio Classification datasets concerning urban sound classification. Amongst others, The DenseNet [10] (Figure 2.2) and ResNet [11] standard models were used for experiments and they have been trained on ImageNet [12], an image database with more than 14 million hand-annotated images. Despite the significant difference between audio Mel-Spectrograms and standard ImageNet image samples, Palinisamy et al. showed that these models can be used as strong baseline networks for audio classification, with DenseNet achieving a validation accuracy of 92.8%. It can be concluded that using pre-trained weights on a CNN with Mel-Spectrogram input is better than using randomly initialized weights.



(Image Source)

FIGURE 2.2: DenseNet201 architecture consists of 6, 12, 48, 32 convolution layers in each of the convolution blocks respectively.

Additionally, Tsalera, Papadakis, and Samarakou [13] compared five CNN-architecture

models : three designed for image classification, and two typically used for sound classification, as they have shown to achieve great results in such tasks. The aim of this experiment was to see which of the above work better for sound training data, after applying transfer learning. The models were trained on three different datasets concerning urban sound classification, where the sound-oriented models outperformed the image-oriented ones, both achieving classification accuracy over 90% on all datasets.

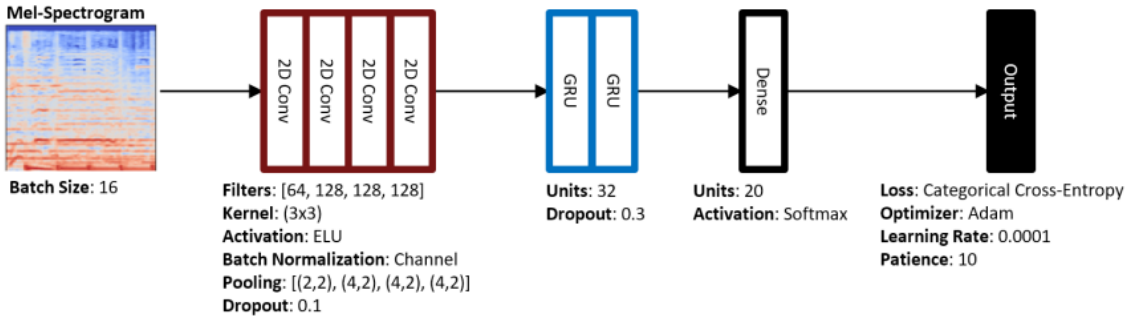
One dimensional Convolution Neural Networks as well as Recurrent Neural Networks (RNNS) are types of deep learning algorithms that can remember sequences. Considering that Mel-Spectrograms are a representation of sound, and sound varies with time, we could assume that a Mel-Spectrogram is a sequence of how the "strength" of the song's frequency is distributed in every time frame. The following papers examine this theory, by testing how a sequential approach to Mel-Spectrograms performs in a deep learning scenario.

Jana, Sharma, and Agrawal [14] proposed a Mel-Spectrogram and 1D-CNN based scheme for Seizure detection. Mel-Spectrograms were then passed through a 3-layer 1-D Convolutional Network and an average accuracy estimated up to 77.57% was achieved. Similarly, Lezhenin, Bogach, and Pyshkin [15] treated Mel-Spectrograms as sequences, due to the temporal structure in audio Mel-Spectrograms. Trained on Mel-Spectrograms extracted from the UrbanSound8K dataset[2] for urban sound classification, the project aimed to capture a song's variation in both time and frequency domains, this time using LSTM [16] cells and achieving 83% accuracy. In comparison to the CNN trained on the same dataset, Lezhenin et al. concluded that the LSTM network had a little performance increase and was more robust. This indicates that in some cases it can be proved beneficial to approach Mel-Spectrograms as sequences, however as of now there is not much research on the topic to establish such an opinion.

Due to the remarkable results produced by CNN models, but also the small but important contribution the RNN architectures seem to have on predictions based on Mel-Spectrograms, some have suggested using a combination of Convolution and Recurrent Layers, using combined deep convolutional and recurrent models in a Convolutional-Recurrent Neural Network (CRNN). Adding GRU [17] or LSTM [16] layers could potentially allow the neural network to pick up on time patterns using what it learns from the convolutional portion.

Nasrullah and Zhao [18] aimed to propose models for music artist classification, where the features regard audio clips, albums and songs of each of 20 artists, taken from a total of 1413 songs. These songs were split into audio clips with lengths varying from 1 to 30 seconds, but also used as a whole, in the form of Mel-Spectrograms. The model was built with both Convolution and GRU layers as shown in Figure 2.3, and it ended up being able to outperform traditional baselines under a range of conditions. The best performing model achieves an average F1 score of 0.937 in recognising the artist.

[2]Dataset from: `https://urbansounddataset.weebly.com/urbansound8k.html`



(Image Source)

FIGURE 2.3: CRNN Architecture for Artist Classification.

An interesting approach was also introduced by Indorf [19], concerning specifically hip-hop tracks. For training, he only used Mel-Spectrograms produced by audio preview samples, and their relative popularity scores. These scores were converted into a binary target (popular / not popular). After training the dataset on both CNN and CRNN architectures, it was concluded that the addition of GRU layers on the initial CNN model gave a peak in accuracy, reaching 59.8%. Lastly, Xu et al. [20] proposed the combination of 1-D Convolution and GRU layers. This paper suggested that using a 1-D CNN for feature extraction from Mel-Spectrograms amongst other low-level audio features, with the addition of GRU based RNNs for the detection of the long-term temporal structure of the audio signal, was able to produce state-of the art results in the domain of audio tagging.

2.2.2 Mel-Frequency Cepstral Coefficients

A different approach to Mel-Spectrograms and audio preprocessing in general has been used broadly for audio classification and regression tasks. This approach concerns the extraction of the Mel-frequency cepstral coefficients from a Mel-Spectrogram, also known as MFCCs. The following papers highlight this method and the benefits of MFCC extraction. Regardless of their objectives being different than popularity prediction, they pin-point some state-of-the art techniques for using MFCCs for prediction.

Habib et al. [21] conducted a deep learning research for assessing the quality of medical consultations based on recordings that were converted into audio and text data. The audio-based part of the project implemented, amongst others, MFCCs and Mel-Spectrograms for training a deep feed-forward neural network. This audio-based approach achieved the highest precision compared to other traditional approaches used up to that point for that objective, concluding that the high-level features contributed to a final precision score of 52%. As a continuation to the implementation of MFFCs features thought, Nawas, Bari, and Khan [22] used a Random Forest Classifier for inferring results on MFCCs and concluded that they have many benefits in showing accurate results and are worthy replacements of other novel features. It is worth noting that, even though the objectives of these papers are different to the one of the current thesis, they both managed to show that MFCCs can achieve just as good results as the Mel-Spectrograms.

Chapter 3

Collected Data

3.1 Spotify Features

Spotify provides high-level features for every song on its platform. They are simple numerical data, abstract and easily understood by humans.[1] The features used in this thesis are summarized in Table 3.1.

Table 3.1: Description of Spotify Features

Spotify Features	Meaning
album_release_date	The date the album was first released. The release dates of the songs in this dataset range from 1982 to 2018.
explicit	Indicates whether or not the track has explicit lyrics.
popularity	The popularity of the track. This is also the target variable.
acousticness	A float measurement of Acousticness. It is a confidence measure, ranging from 0.0 to 1.0, of whether the track is acoustic or not. 1.0 represents high confidence the track is acoustic.
danceability	It describes how suitable a track is for dancing, taking into account several factors such as tempo, rhythm, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

Source: Spotify Web API

Continued on next page

[1] See https://developer.Spotify.com/documentation/web-api/reference/#/operations/get-audio-features for more information.

Table 3.1: Description of Spotify Features (Continued)

Spotify Features	Meaning
instrumentalness	Measurement of the likelihood the track is instrumental - whether it contains vocals or not. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
key	The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C/D, 2 = D, and so on. If no key was detected, the value is -1.
liveness	Measurement of the likelihood the track is live.This value describes the probability that the song was recorded with a live audience. Higher liveness values represent an increased probability that the track was performed live.
loudness	Overall loudness of a track compared to other Spotify tracks, in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 db.
mode	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
speechiness	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
tempo	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of given piece and derives directly from the average beat duration.
time_signature	The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".

Source: Spotify Web API

Continued on next page

Table 3.1: Description of Spotify Features (Continued)

Spotify Features	Meaning
valence	A measure from 0.0 to 1.0 indicating the positiveness of a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

Source: Spotify Web API

The correlation matrix between those features is visualized in Figure 3.1, indicating which of them are the most correlated. The variables *energy* and *loudness* have the highest correlation. This correlation is positive, which indicates that the louder the song is, the most energy it has. Additionally, *acousticness* and *energy* are negatively correlated, since more acoustic songs are usually calmer and tend to have less energy. This also explains why *loudness* also has a highly-correlated inverse relationships with *acousticness* and *instrumentalness*. The target variable *popularity* seems to have some small correlation with the variables *loudness*, *instrumentalness*, *explicit* and *danceability*, which suggests that these features might be significant to its prediction.

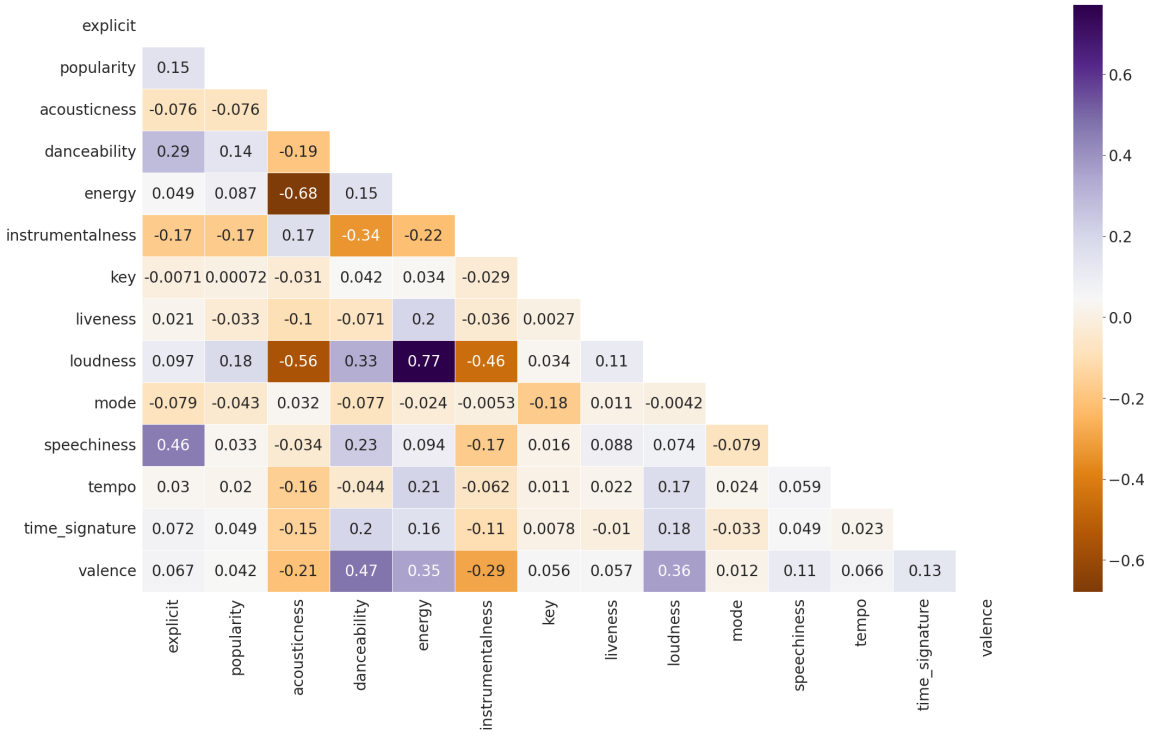


FIGURE 3.1: Correlations between Spotify Features

Although there are millions of well-written, well-composed songs, the majority of them are not hits. Indeed, wonderful songs are constantly produced but they won't always succeed; talented musicians don't always achieve success commensurate with their abilities. This makes predicting hits a difficult task, as many promising songs will not be hits regardless of how good they might actually be, as there are many exogenous variables

influencing its success such as the promotion done from the record label or the previous popularity of the artist. Generally, popular songs a minority in the music industry.

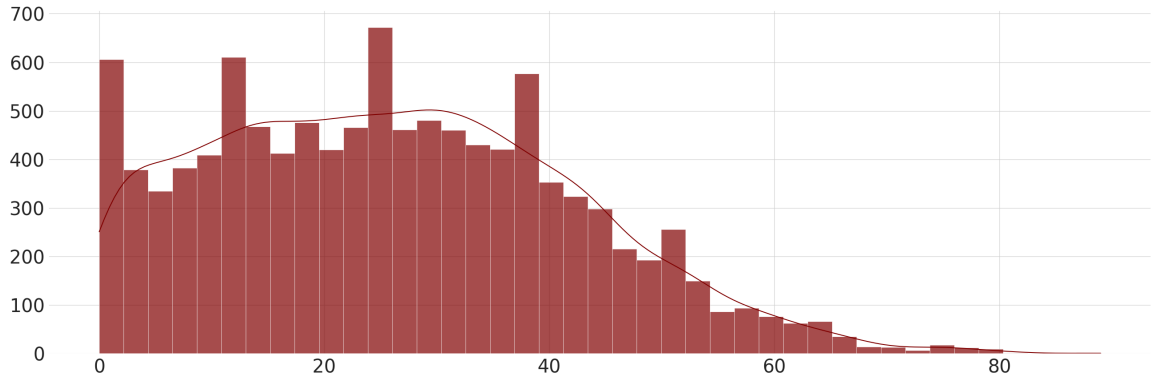


FIGURE 3.2: The right skewed popularity distribution shows how rare it is to have a popular song.

The theory is also supported by the given dataset; according to Figure 3.2 the number of songs seems to decrease as the popularity score rises. We try separating the songs in three equal-width categories: songs of low, medium and high popularity.

- **[0, 29]:** If the song popularity score is in that range, it is classified as of low popularity. 59% of the songs belong in this category.
- **[30, 58]:** If the song popularity score is in that range, it is classified as of medium popularity. 39% of the songs belong in this category.
- **[59, 89]:** If the song popularity score is in that range, it is classified as of high popularity. Only 2% of the songs belong in this category.

Assuming that a truly successful song has a score over 59, an inference that sounds sensible after the consultation of Figure 3.2 is that hit songs are very under-represented in the provided dataset, and their detection is bound to be challenging.

3.2 Genres

In addition to features from Spotify, two additional features concerning Genre are provided: genre and subgenre. A song can have multiple genres and subgenres. For each song, one to four words are used to indicate its genre(s) and subgenre(s) respectively. Most songs do not have a subgenre. Figure 3.3 shows the most common genres in the dataset, by using the first (and in some cases, only) word of the song genre(s). From this visualization, it seems that the Pop genre is the most represented, followed by Alternative and Rap. The genres that do not appear on this visualization are assigned to less than 100 songs in the total 10k sample dataset.

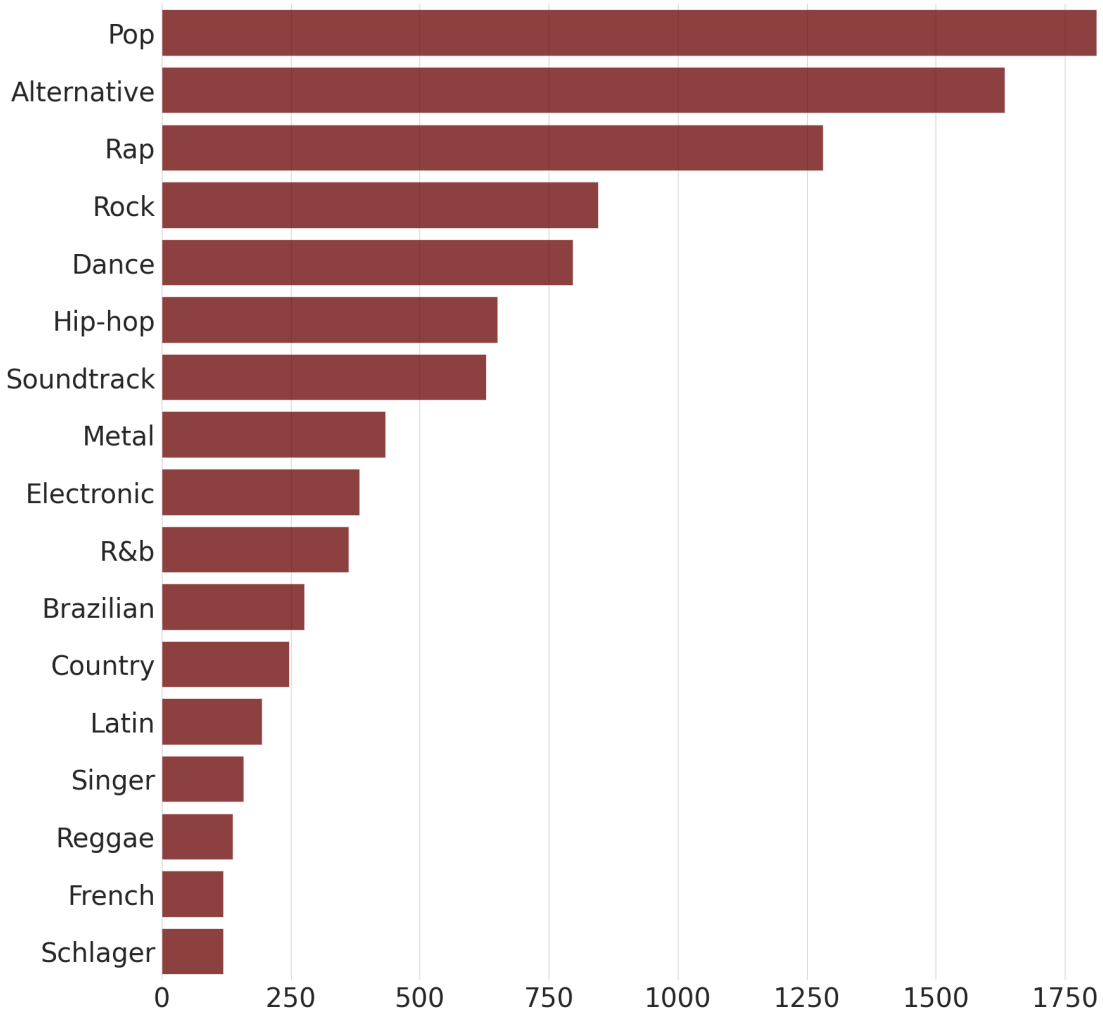


FIGURE 3.3: Number of Songs Per Genre

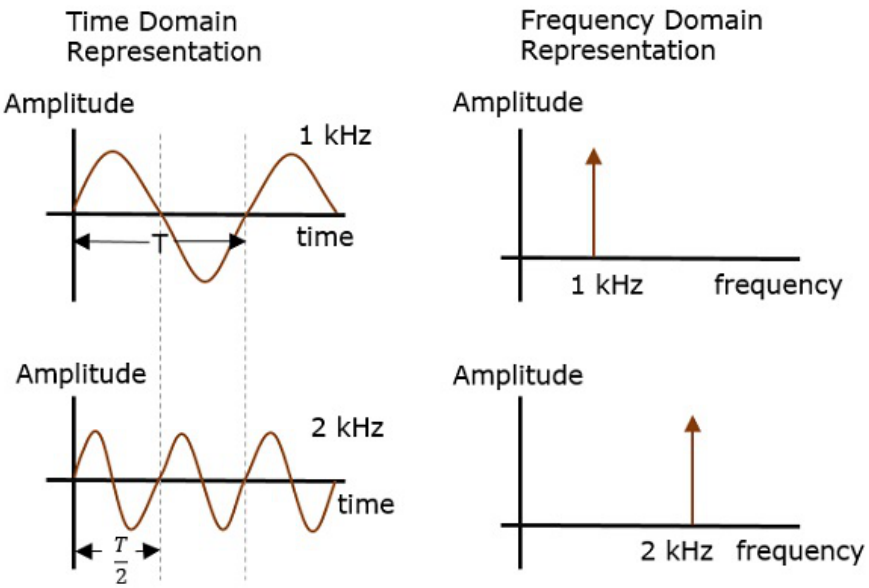
3.3 Mel-Spectrograms

The computer only understands numerical input. This means that sound has to be represented digitally in order for information about a song's audio to be efficiently extracted, especially in a deep learning scenario. Mel-Spectrograms are the most common approach to represent sound in deep learning scenarios, as they are an easy way to transform a sound into an image, and images can be comprehended by Neural Networks. However, since they are low-level audio features which are solely obtained from audio, they have to go through processing to be ready to be taken as input.

3.3.1 Audio Digital Representation

Sound is produced by vibrations in air pressure. As mostly studied in the field of physics, these vibrations create a wave, as sound travels through space. A wave can be used as a way to both visualize and measure information about the sound, such as its frequency and intensity, also known as amplitude.

Audio signals are the primary representation of sound. An audio signal can tell us how sound is distributed in fixed intervals of time, measuring its "intensity" by its amplitude. Having the audio signal of a sound is sufficient to reproduce it. By looking how sound is distributed by time, we view the signal in the *Time Domain*. Signals of different frequencies can be added together, thus representing more complex patterns in sound. When the sound is distributed by frequency, the signal is viewed in the *Frequency Domain*, which is also called the spectrum of a song. These two representations can be seen in Figure 3.4.



(Image Source)

FIGURE 3.4: Signal Representation in both Time and Frequency Domain

A song is a complex sound, which can be produced by any number of accumulated sounds or even by complex interactions between sound waves. Just like a signal, these sounds also

vary with time. Intuitively, we want to break down this complex sound into its frequency components for the duration of a song. The first step of this process is to divide the audio signal of the song into short overlapping windows. An audio signal is constantly changing, so we use this method in order to to simplify things by assuming that on short time scales the audio signal is statistically stationary. The next step is to calculate the power spectrum of each frame, which basically identifies which frequencies are present in the frame and at what intensity. This technique applies a short-time Fourier transform (STFT) to convert the signal from the *Time Domain* to the *Frequency Domain*. The mathematical form of the discrete STFT is shown in Equation 3.1:

$$\mathrm{STFT}\{x(n)\}(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]\omega[n-m]e^{-i\omega n} \tag{3.1}$$

where $x[n]$ and $\omega[n]$ denote the input signal and the frequency content in short-window frames of audio. The variable m is a time argument and the variable n identifies the location of the short segment of the original time function as it is extracted using the window $\omega[n-m]$, which moves along the m-axis according to the value of n.

Applying Fourier Transformations to time segments of a song and combining them together produces a spectrogram. A spectrogram is the most compact visual representation of an audio signal in terms of the signal's loudness or intensity as it shows the variation of the frequencies of acoustic signals over time. The frequency distribution at each time instance of a sound is efficiently described by displaying the amplitude of each frequency present in the signal. However, a spectrogram alone does not give sufficient information about the song. As seen in Figure 3.5, the amplitude differences in both lower frequencies and higher frequencies are not easily detectable, as the colors are quite similar. In order to extract useful information from the spectrogram, it needs to be slightly modified.

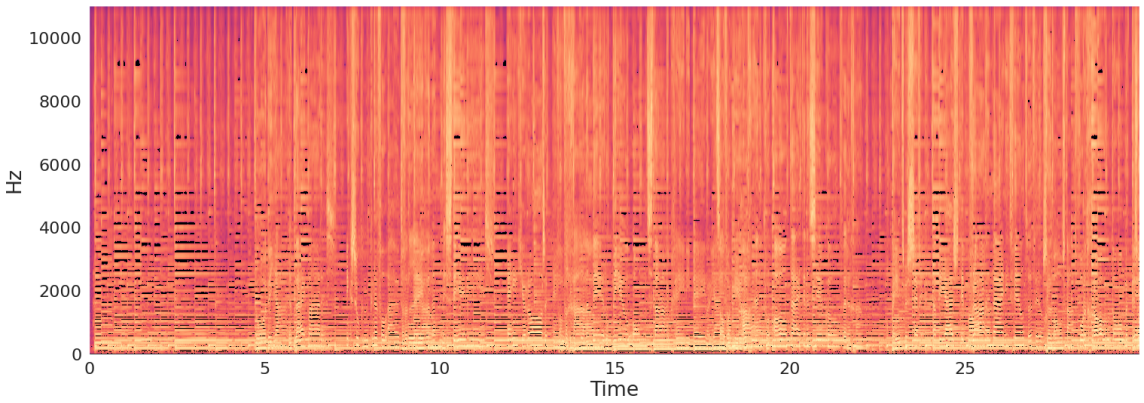


FIGURE 3.5: Spectrogram of a song

3.3.2 The Mel Scale

The human brain perceives frequency logarithmically, whereas simple spectrograms represent it linearly. The Mel-scale, as described by Stevens, Volkmann, and Newman [6], is "*A scale of pitches judged by listeners to be equal in distance one from another*" and it aims to

mimic the non-linear human ear perception of sound, by being more distinctive at lower frequencies and less distinctive at higher frequencies [6]. In contrast to the simple spectrogram, the frequency scale (f, Hz.) is now transformed into the Mel scale (m, Mels) through Equation 3.2:

$$m = 2595 \cdot \log_{10}(1 + \frac{f}{700}) \tag{3.2}$$

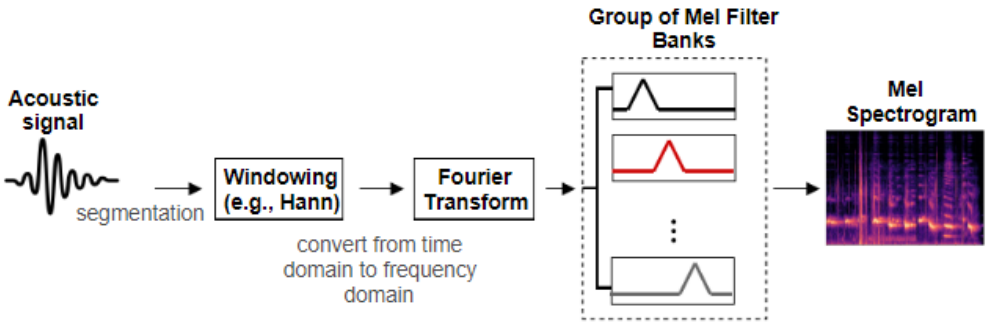
and then scaled to decibels (d) through Equation 3.3, where r is a reference quantity used to transform the ratio of both m and r into the log-domain:

$$d = 10 \cdot \log_{10}(\frac{m}{r}) \tag{3.3}$$

Mel filter banks[2] are used to decompose the audio signal into separate frequency bands in the mel frequency scale, by being non-uniformly placed in the frequency axis to simulate human ear properties. Therefore, the filters are more centered around the low-frequency region where the human ear cannot distinguish differences between two closely spaced frequencies and less in the high-frequency region. A Mel-scaled power spectrogram of a signal is finally produced by applying the Mel-scaled filters to the power spectrum of a signal, as calculated by Equation 3.1, and the logarithm of the energy output of each filter. This can be expressed as:

$$S[m] = \log_{10} \sum_{k=0}^{N-1} (|x[k]|^2 \cdot H_m[k]) \tag{3.4}$$

where $H_m[k]$ are the filter-banks, and m is the number of the filter-bank. Practical Cryptography [23] offers a more detailed explanation of all the processes and computations above. These processes are also summarized in Figure 3.6.



(Image Source)

FIGURE 3.6: Process of extracting the Mel-Spectrogram from an acoustic signal.

[2] A filterbank is basically a way to discretize a continuous frequency into bins.

In Figure 3.7, the Log-Power Mel-Spectrogram of the same song as above is produced. Here, the x-axis denotes the time segments of a song (seconds) and the y-axis shows the mel-frequency (to be precise, the center frequency of each mel band in Hz). Each pixel-point in the Mel-Spectrogram, being represented by color, tells us how present a frequency is at each point in time using Decibels as a measure of intensity.

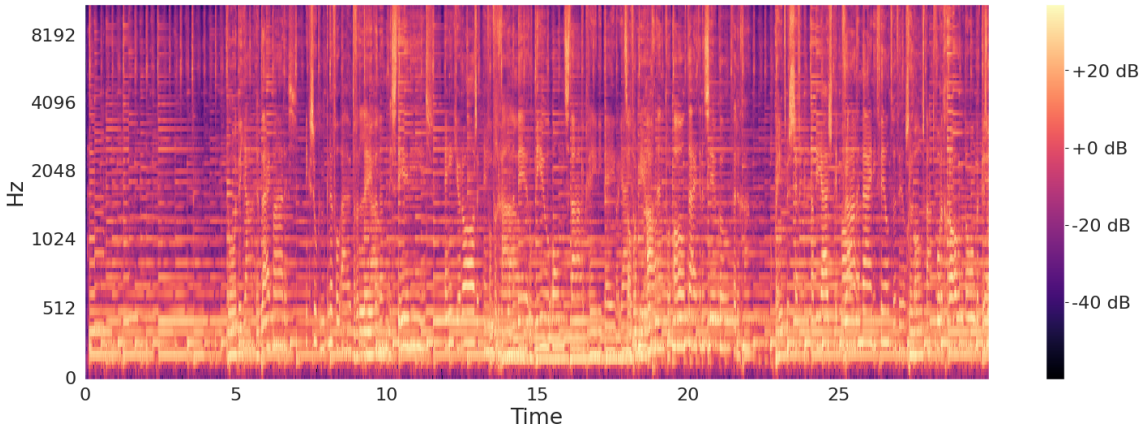


FIGURE 3.7: Log-Power Mel-Spectrogram of a song

3.3.3 Mel-frequency Cepstral Coefficients (MFCCs)

A term that very often is mentioned along Mel-Spectrograms is Mel-frequency Cepstral Coefficients (MFCCs). Widely used in Automatic Speech Recognition (ASR) tasks, MFCCs are a compact representation of the spectrum of an audio signal and are created by reducing the frequency information of the signal into a small number of coefficients. The process of extracting the MFCCs consists of several mathematical transformations of the signal similar to the ones used for the Mel-Spectrogram such as the Fourier transformation, as well as the Discrete Cosine Transformation (DCT) to obtain a log-magnitude representation of the spectrum. The DCT is applied on the logarithmed filter banks, which are computed as described in Equation 3.4. This results in the cepstral coefficients as represented by:

$$c_i = \sum_{n=1}^{N_f} \log_{10}(s_n) cos(\frac{\pi i(n-0.5)}{N_f}), i = 1, 2, \ldots, L \tag{3.5}$$

where c_i is the i-th MFCC coefficient, N_f is the number of triangular filters in the filter bank, S_n is the log energy output of n-th filter coefficient and finally L is the number of MFCC coefficients that we want to calculate. MFCCs are also visualized in Figure 3.8.

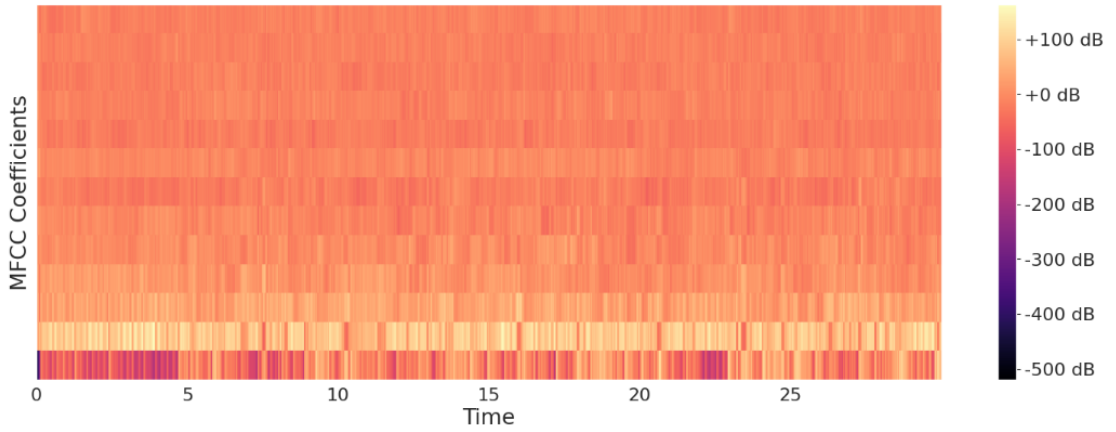


FIGURE 3.8: 13 Mel-frequency Cepstral Coefficients of a song

Conventionally, the MFCCs are from 8 to 13 features. However, these features only offer information about single time-frames. The generation of additional temporal features is done by finding the first and second derivatives of the cepstral coefficients called Delta and Delta-Delta features. Also known as Differential and Acceleration coefficients, these features are used to represent the temporal information by introducing even longer temporal context and have shown to increase performance quite a bit. The reason this is done is because calculating these MFCC trajectories and appending them to the original feature vector has shown to increase model performance especially in ASR tasks. The Delta coefficients are calculated as:

$$d_t = \frac{\sum_{n=1}^{N} n\left(c_{t+n} - c_{t-n}\right)}{2\sum_{n=1}^{N} n^2} \tag{3.6}$$

where d_t is a Delta coefficient from frame t, computed in terms of the static coefficients c_{t+n} to c_{t-n}. N is usually equal to 2 and denotes that the previous and next frame are used. Delta-Delta coefficients are calculated in the same way, but they are calculated from the Deltas.

3.3.4 Mel-Spectrograms vs MFCCs

A spectrogram is able to distinguish each of the sound elements in an audio recording, and represent the song structure with great detail, in visual format. This format can be used as input in many different deep learning scenarios, such as Convolution Neural Networks (CNN) and Recurrent Neural Networks (RNN). On the other hand, MFCCs are often preferred because they are more easily compressible, decorrelated and they take up less memory. There are many python libraries, with the most commonly used being librosa[3],which is specifically created for audio processing. Raw audio data, such as a .wav file can be converted to spectrograms or MFCCs through this library, according to the operations mentioned above. These operations are considered some of the most standard practices for audio processing, since they have been shown to achieve impressive results.

[3]See https://librosa.org/doc/latest/index.html for more information

3.4 Youtube Views

In addition to Spotify features, Genres and Mel-Spectrograms, YouTube views from the years 2019 to 2022 were given for each song. Starting from January 2019 up until May 2022, we have information for total and peak views in the music video of the song per month. Using the number of views for every song can give a very significant peak to the model's performance, as it is a clear indicator of how well a song is performing up until the point that we need to predict its popularity.

Total views are the total amount per views the music video of a song gets every month (Figure 3.9). According to YouTube, these include all of the times videos featuring the song recording were played on YouTube or YouTube Music, including the official music video, the audio track, and all fan videos like lyric videos.[4] Similarly, peak views (Figure 3.10) are the highest number of views in a day per month.

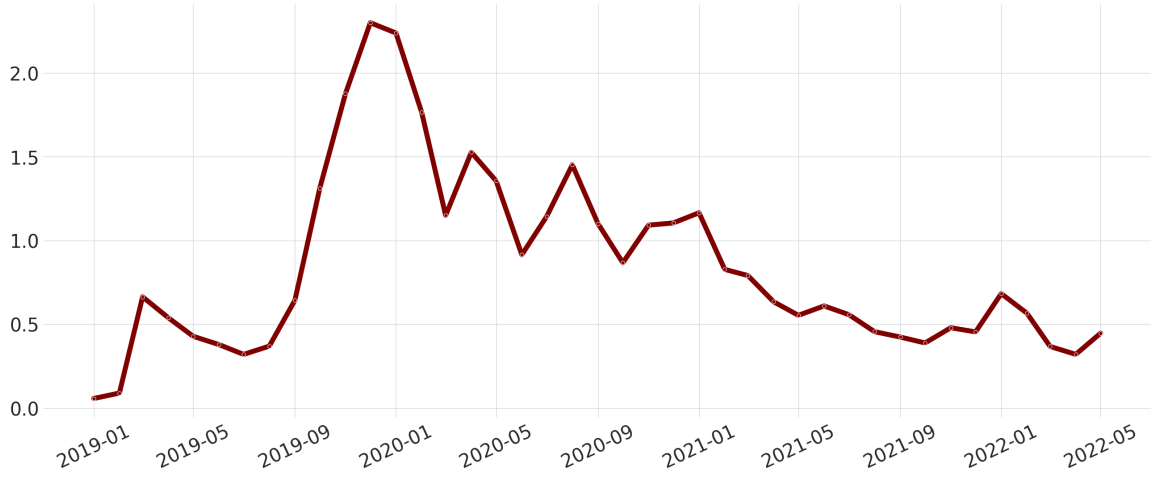


FIGURE 3.9: Total Views (in hundreds of millions) of a song per month from years 2019-2022

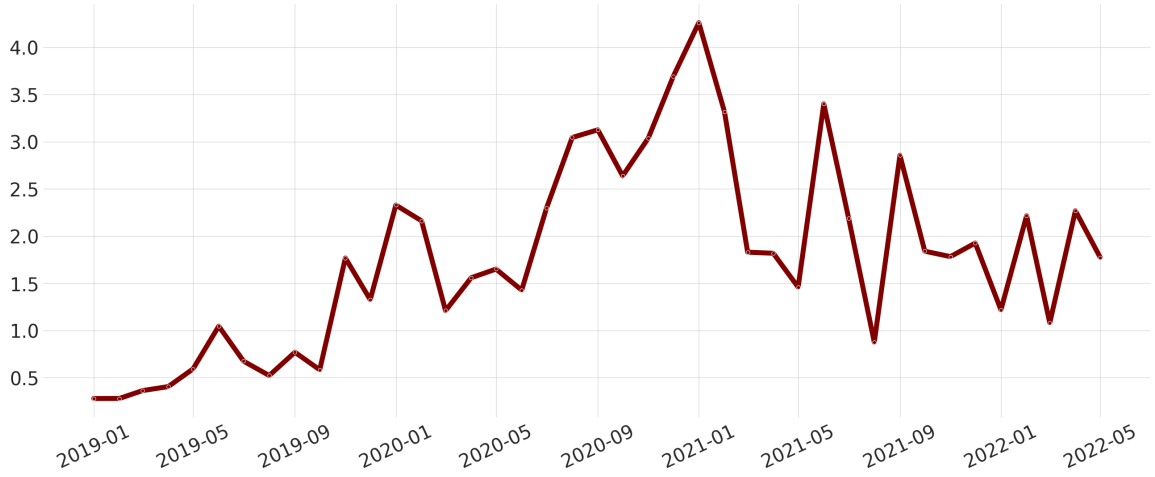


FIGURE 3.10: Peak Views (in tenths of millions) of a song per month from years 2019-2022

[4]Source: https://support.google.com/youtube/answer/9419340?hl=en

3.5 Data Preprocessing

- **Spotify Features**

As mentioned above, each song has 15 Spotify Features that describe it. Nasreldin et al. [1] suggested that using the release date of the song could potentially increase model performance. For that reason, the year and the month of the *album_release_date* feature are added as features, and the *album_release_date* feature is dropped.

As far as categorical values are concerned, two variables are already in binary format (*explicit* and *mode*). Certain categorical variables, such as *key*, are value-encoded, but they do not have any particular substance or order. If 0 is the key of C, and 1 is the key of C#, this does not mean the key of C# is intrinsically greater by 1 point than the key of C. Additionally, *key_signature* is an already predicted variable, and its relative values indicating beats per bar on a song are also meaningless. These two variables are converted to Boolean features, increasing the total number of features by the number of unique values for each variable.

- **Genres**

The Genres of each song are text features and as a result are processed differently. Firstly, the text is "cleaned-up": insignificant characters such as commas and backslashes are removed, with the exception of "-" and "&" as they are important characters in genres like Hip-Hop, R&B etc. The first letter of every word is lowered and similarly worded genres are replaced with a new shared name. For instance, the initial provided genre *['Rap/Hip-Hop', 'Hip-Hop']* of a song is now just *rap hip-hop*. Additionally, two new features are created: *all_genre*, which is the combination of genre(s) and subgenre(s) in one feature, and *first*, which is essentially the first word in a song's genre, as it could be assumed to be the most important one. The goal of creating all these additional features is to further explore which of them has the greatest predictive strength when it comes to popularity.

- **Mel-Spectrograms**

By treating Mel-Spectrograms as images, we can borrow from the many powerful ideas in image recognition with deep learning. A spectrogram, however, is fundamentally different than natural images. As mentioned above, Mel-Spectrograms are used to provide the models with sound information similar to what a human would perceive. This representation allows us to utilize spectrograms using well documented image processing techniques, as well as approach them as sequences of feature vectors through time. In order to experiment with spectrograms in these two deep learning scenarios, their values are converted to range from 0 to 1, because the computation of high numeric values may become more complex for a Neural Network and the following experiments showed that when values are in this range, models perform better. In the image approach, the values are first converted to range 0 to 255, as to represent an image, and then divided by 255.

Mel-Spectrograms have been proven to perform greatly on deep learning scenarios. However, **MFCCs** offer a more compact representation of low-level audio features and contain

features that can be converted in tabular format, given that way as input to Machine Learning models. It is important to mention that each coefficent is essentially a vector of values. As mentioned above, in practice the first 8–13 MFCC coefficients are used. For this thesis, 13 MFCC coefficients will be used. The Delta coefficients are calculated by the MFCC coefficients and in turn the Delta-Delta coefficients are derived from the Delta's. As a result all coefficients have 13 features each and accordingly, the combined MFCCs features are extended from 13 to 39 feature vectors. In order to compress these feature vectors and then transform them to tabular format, the mean is taken across every coefficient, so that now every coefficient is represented by a number. This is generally a common practice in such tasks. As a result, a song is now described by a 39-Dimensional vector, each of its values deriving from a coefficient feature vector. From experiments made in the given dataset, it was concluded that scaling these data by removing the mean and dividing by the variance has the best performance.

- **YouTube Views**

YouTube Views are used in two different ways: by extracting static features as well as in Multivariate Time Series scenario. The decision to extract features was made due to the fact that the data for every song concern only 41 months, and may not be sufficient for prediction in a Neural Network. Due to the fact that 40 observations is often mentioned as the minimum number of observations for a time-series analysis [24], it was decided that the extraction of static features from these time-series could potentially provide a better insight into how YouTube views contribute to the overall popularity of the song, by exploring their trend, scale etc.

The feature extraction was made using tsfresh[5], which is an automated feature extraction and selection library for time series data. Through feature calculators, these features include computations that range from simple analytics to autocorrelation and quantiles. More than 750 features can be extracted from each time series, which are a bit too many comparing to the 15 total features obtained from Spotify. For that reason, a much smaller number of features was extracted. Additionally, four more features are created concerning the coefficients of the time-series line of each song. Performing a simple Linear Regression, the "trend" for every song of all time, the last 12 months and the last 6 months is extracted for Total Views (Figure 3.11) and Peak Views (Figure 3.12) respectively.

[5]See `https://tsfresh.readthedocs.io/en/latest/index.html` for more details.

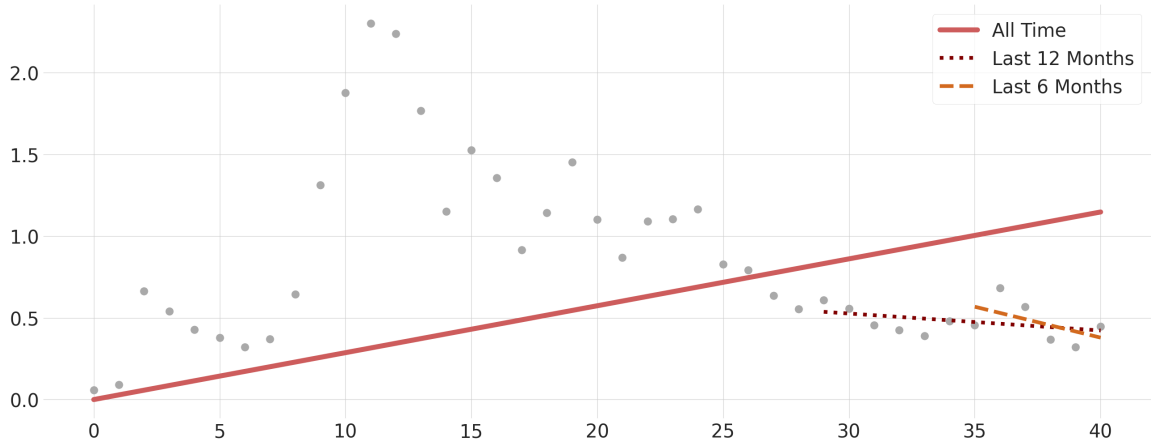


FIGURE 3.11: Visualization of the Trend Lines of Total Views (in hundreds of millions) of a song from years 2019-2022

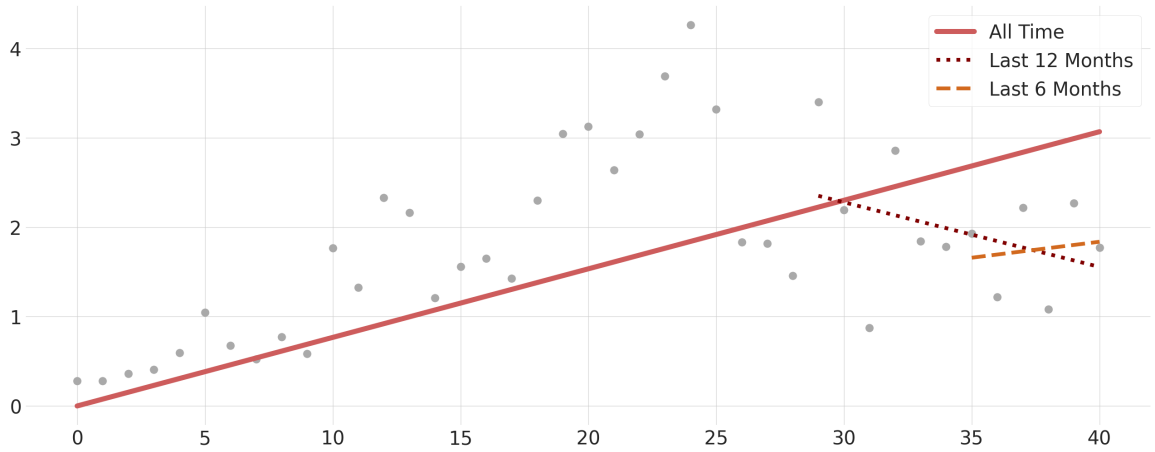


FIGURE 3.12: Visualization of the Trend Lines of Peak Views (in hundreds of millions) of a song from years 2019-2022

Feature selection using mutual information is applied to select the 20 best features of the above.[6] The final features representing time series are presented in Table 3.2.

[6]The implementation of sklearn was used. See https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html for more details.

Total Views	**Peak Views**
Sum of Values	Sum of Values
Median	Median
Mean	Mean
Standard Deviation	Standard Deviation
Variance	Variance
Root Mean Square of observations	Root Mean Square of observations
Maximum	Maximum
Absolute Maximum	Absolute Maximum
Minimum	
All Time Coefficient	All Time Coefficient
Last Year Coefficient	

TABLE 3.2: Static Features for YouTube Views

The initial time-series data are scaled for every song by removing the mean and scaling to unit variance.[7] Finally, they are converted in a Multivariate Time Series format: each song is described by two 41-Dimensional vectors; one for Total Views and one for Peak Views. These vectors are combined in a single 41x2 array for every song, where each element of the vector represents the standardized total and peak views respectively in chronological order.

All datasets respectively undergo a 80/10/10 split to form the training, validation and testing sets. The splitting is made according to the songs' chronological releases; older songs are kept in the training set, and newer ones are used for validation and testing. All the scaling mentioned above is applied separately in the training, validation and testing sets, in order to avoid information leakage. Songs of all popularities are evenly distributed in all sets, meaning that the initial analogy of approximately 59% of low-popularity songs, 39% of medium-popularity songs and 2% of high popularity songs is preserved in every set.

The imbalanced nature of this dataset is also taken into consideration and the popularity score is ensured to be equally accounted for in each set, in order to avoid having songs of unrepresented popularity be "overshadowed" by the ones that are of more frequent popularity. As seen in Figure 3.2, the provided dataset includes more songs that are not considered hits rather than songs that are. Consequently, songs with the highest popularity scores must be given weight so as to subtly influence the model to detect the future songs that more probable to become hits. For that reason, custom **Sample Weights** will be used, converting initial weights from 1 to a new value that occurs according to their frequency on their dataset, giving higher weights to songs whose popularity is less frequent on the dataset. Data Augmentation was also considered but eventually could not be applied, as the data used for prediction are of much different formats and artificial data for the totality of them could not be created.

[7] The implementation of sklearn was used. See https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html for more details.

Chapter 4

Methods

In this section, the models and the methods used for the task of popularity prediction are discussed. These methods range from Tree based models to Convolutional and Recurrent Neural Networks.

4.1 Random Forest

A Decision Tree [25] can be defined as supervised learning method and it follows a flowchart-like tree structure. The aim of a tree is to predict the value of a target variable by learning simple decision rules inferred from the data features. Every decision tree has high variance, but by combining multiple of them we can achieve a reduced variance with a slight risk of overfitting due to the consequent increase in bias. This is the basic idea behind Random Forests [2]; to combine diverse decision trees in order to determine the final output rather than relying on individual decision trees.

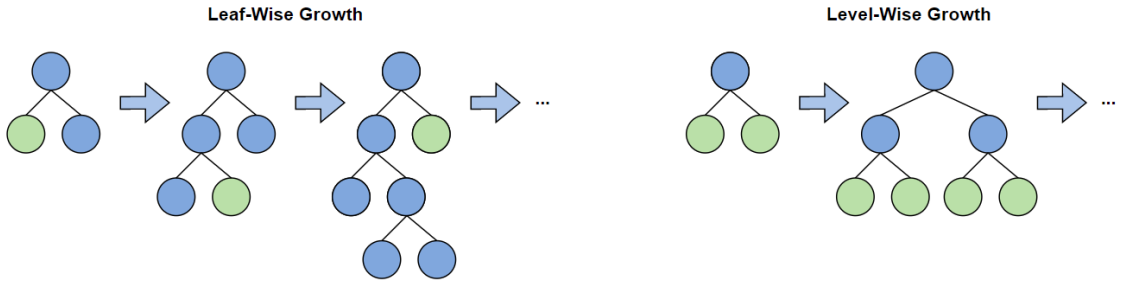
4.2 XGBoost

XGBoost [4] stands for eXtreme Gradient Boosting and is an implementation of Gradient Boosted decision trees. Gradient Boosting is a popular methodology in machine learning in which an ensemble of weak learners is used to improve the model performance, with the aim of finally combining their outputs to get better overall results. In the XGBoost algorithm, decision trees are created in sequential form, each trying to improve its predecessor's performance by assigning a higher weight to observations it failed to predict correctly. Finally, these decision trees are ensembled to create a stronger model. XGBoost is a very popular model and often outperforms complex models such as neural networks, due to its execution speed and performance.

4.3 LightGBM

LightGBM [26], which stands for Light Gradient Boosting Model, is a gradient boosting framework based on decision trees originally developed by Microsoft. This algorithm basically implements the conventional Gradient Boosting Decision Tree (GBDT) with the addition of two techniques (Gradient-based One-Side Sampling and Exclusive Feature

Bundling), which are designed to significantly improve the efficiency and scalability of GBDT. In contrast to XGBoost where trees grow depth-wise, in the LightGBM trees grow leaf-wise. This architecture results in more loss reduction and in turn higher accuracy while being faster, but it can often be prone to overfitting. XGBoost and LightGBM generally yield similar performance, with the latter being much faster.

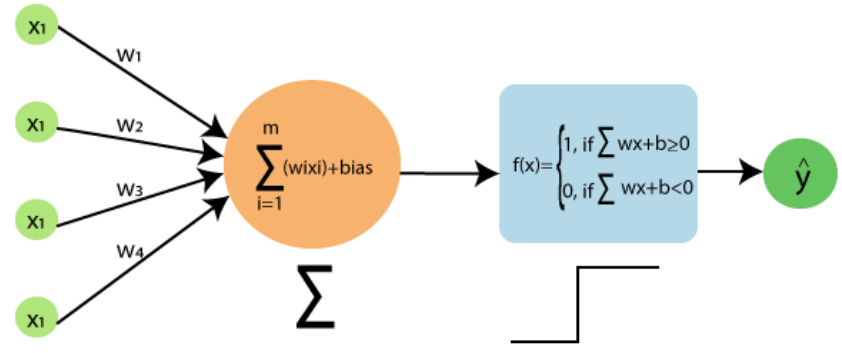


(Image Source)

FIGURE 4.1: LightGBM (left) vs. XGBoost (right)

4.4 Multi-Layer Perceptron

An Artificial Neural Network (ANN) tries to recognize patterns and solve common problems in the fields of Artificial Intelligence, Machine Learning, and Deep Learning. The perceptron is a linear classifier, used to classify its input into one or two categories. As seen on Figure 4.2, it consists of 4 parts: the input layer, which takes the initial features into the network, the activation function and the output, the weights and bias, which represent the "strength" of the connection between units with the addition of a constant value for every input, the summation of weight and bias for every input and finally the activation function which calculates a weighted sum to produce the output.



(Image Source)

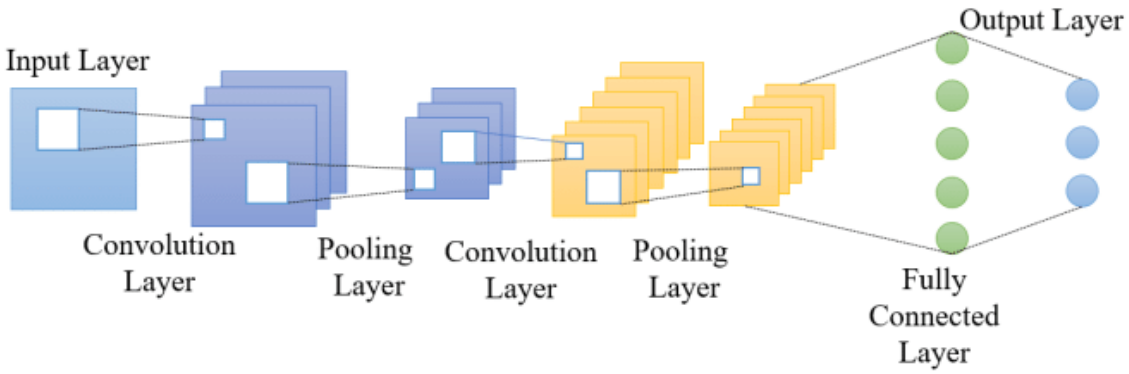
FIGURE 4.2: Single Layer Perceptron

A Multi-Layer Perceptron (MLP) is a feedforward ANN that generates a set of outputs from a set of inputs and is substantially formed from multiple layers of the perceptron. It consists of at least three layers of nodes: an input layer, a hidden layer and an output layer and uses backpropogation for training the network. The MLP learning procedure follows

three basic steps. At first, the data are propagated forward from the input layer towards the output layer. Then, based on the output, the difference between the predicted and true value is calculated and is denoted as the error of the model, which needs to be minimized through backpropagation. The derivatives of the error with respect to each weight in the network are computed, and the model is updated. These steps are repeated a pre-selected amount of times (epochs) in order for the ideal weights to be learned.

4.5 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are most commonly applied to computer vision. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and even natural language processing. In contrast to simple Multi Layer Perceptrons (MLPs) that operate directly on raw observations, CNNs are able to minimise human effort by automatically detecting useful features from raw data such as images, offering much greater performance.



(Image Source)

FIGURE 4.3: Example architecture of a 2D-CNN

As shown in Figuure 4.3, there are three types of layers that make up the CNN. These are Convolutional, pooling and fully-connected layers. The term "Convolution" denotes the mathematical function of convolution applied to the input, the result of which is then passed to the next layer. Pooling layers, also known as downsampling layers, basically conduct dimensionality reduction, by summarizing the presence of features in smaller patches of the feature map. When these layers are stacked, a CNN architecture is formed. Usually after the CNN part of the model is complete, a flatten layer is used for reducing the pooled feature maps into a one-dimensional continuous linear vector, to be given as input to the fully-connected layer. Finally, the fully-connected layer interprets the features extracted by the Convolutional part of the model for the final predictions, whether these be classification or regression outputs. One of the big advantages of CNNs is parameter sharing, a scheme used to control the number of parameters by having all neurons share the same weights in a particular feature map. This helps reduce the number of parameters

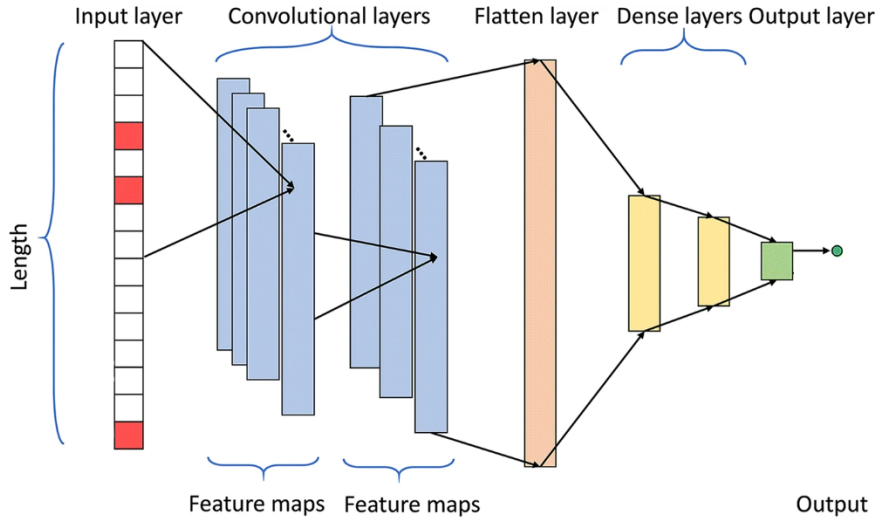
in the whole system and makes the computation more efficient by reducing the training time.

Pre-trained models are very often used for tasks such as image classification and have been shown to be a technique with great advantages. Instead of initializing the model with random weights, initializing it with pre-trained weights can reduce the training time and hence is more efficient. While it may be argued that Mel-Spectrograms are much different than the images these models were trained on as they do not represent objects but are rather just representations of sound, great results have been achieved with Mel-Spectrograms regardless as mentioned in [13].

The DenseNet-201 [10] model was applied, which is a Convolutional Neural Network that is 201 layers deep. In a DenseNet architecture, each layer is connected to every other layer, hence the name Densely Connected Convolutional Network. The input of each layer inside DenseNet is the concatenation of feature maps from previous layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters. Additionally, the VGG-16 [27] model was also applied for comparison as it is a far less complex model - only using 16 layers. VGG-16 is a Convolutional Neural Network trained on 1.2 million images to classify 1000 different categories. Due to the different nature of the Mel-Spectrogram image, transfer learning was also applied.

A two dimensional convolution layer learns a fixed number of local patterns and tries to match it everywhere in the input. This is very efficient for images where all pixels have the same "meaning" and one pixel in an image can most often be assumed to belong to a single object. However, visual objects and sound events do not accumulate in the same manner. In Mel-Spectrograms different locations have different meanings pixels at the upper part of the Mel-Spectrograms indicate high frequencies while the lower part indicates low frequencies. Two dimensional Convolutional Neural Networks use two-dimensional filters that share weights across the x and y dimensions, but in Mel-Spectrograms the two dimensions represent fundamentally different units, one being frequency and the other being time. Therefore, matching a local region in the Mel-Spectrogram may mean a completely different thing if it is matched to the upper or lower part of the image, alas vertical positioning is important in Mel-Spectrograms. Moreover, moving a sound event horizontally offsets its position in time and it can be argued that a sound event means the same thing regardless of when it happens. One dimensional Convolutional Neural Networks can be used to detect features in a vector, making them a great approach to all types of sequential data, such as Mel-Spectrograms and time series. An example architecture of a one dimensional convoltuional network is shown in Figure 4.4.

A particular observed frequency in a Mel-Spectrogram cannot be assumed to belong to a single sound. As previously explained in section 3.3.1, this frequency can be made up of many different, added up sounds as well as their interactions. Sounds do not exist as static objects which can be observed in parallel, they arrive as sequences of air pressure and meaning about these pressures must be established over time. This is in contrast to the usual way 2-D CNNs work, where similar neighboring pixels can be assumed to belong



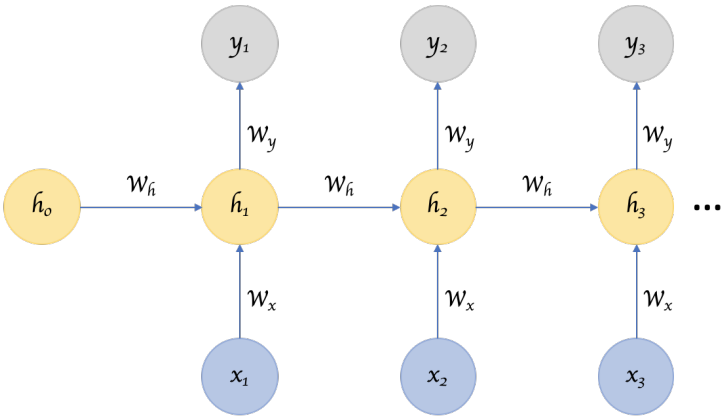
(Image Source)

FIGURE 4.4: Example architecture of a 1D-CNN model. This model consists of two Convolutional layers, one flatten layer, two dense layers and one output layer

to the same visual object; in sound, frequencies are most often non-locally distributed on the Mel-Spectrogram.

4.6 Recurrent Neural Networks

Recurrent Neural Network (RNN) is an Artificial Neural Network whose architecture is specialized for processing sequential data (Figure 4.5). RNNs are known as the "memory" networks - they have the ability to remember the past and its decisions are influenced by what it has learnt from the past. They are mostly utilized for time series and natural language processing tasks due to their ability to take information from prior inputs that will in turn influence the current input and output.



(Image Source)

FIGURE 4.5: A Recurrent Neural Network, with a hidden state h_t that is meant to carry pertinent information from one input item in the series to others.

In contrast to other traditional models, that treat inputs and outputs as independent of each other, the output of the RNN strongly depends on the prior elements within the sequence. This is done by using a hidden state vector h_t representing the context based on prior inputs x_t and outputs y_t. Its aim is to capture the relationship that neighboring inputs might have with each other and it keeps changing in every step. So, the same input could produce a different output depending on previous inputs in the series. As mentioned above, h_t is calculated based on the previous hidden state and the input at the current step by the following formula:

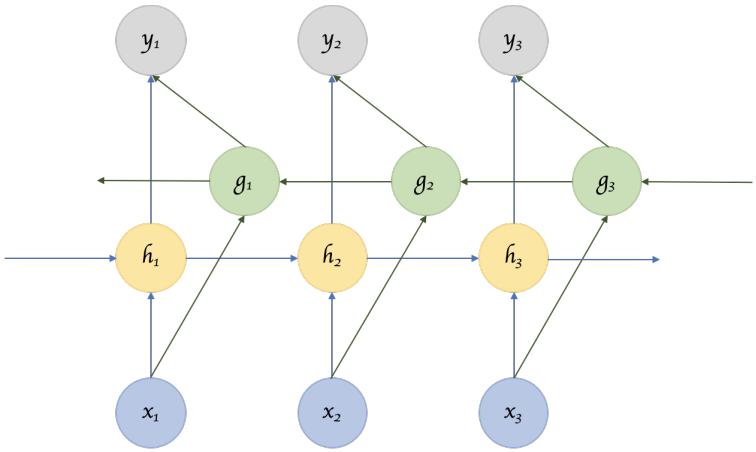
$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h) \tag{4.1}$$

where σ_h is the activation function, W_h and U_h are weight matrixes and b_h is a bias term.

Each hidden stage h_t shares the same weights W_h. This parameter sharing helps in getting rid of limitations caused from applying the model to training examples of different lengths (e.g. sentences in natural language processing do not have the same size). In that way, each different input node does not require weights of its own. The great performance of both CNNs and RNNs can be attributed to the concept of "parameter sharing", which is fundamentally an effective way of taking into account the relationship between one input item and its surrounding neighboring inputs.

4.6.1 Bidirectional RNNs

Recurrent Neural Networks focus on capturing the sequential information at time t by using historical states only. However, sometimes it's important not to only learn from the past to predict the future, but also look into the future to fix the past. Knowing what's coming next helps in better understanding the context and detect the present. As a result, it could be proven beneficial to allow the RNN model to learn the input sequence both forward and backward. Indeed, in a song both back and forward sequence is quite significant, as the timing of either e.g. the bridge or chorus, could be the reason this song becomes a hit. As shown in Figure 4.6, Bidirectional RNNs do just that - they train the model from both directions using two RNN layers, with one (denoted by h_t) moving forwards from start to end and the other (denoted by g_t) moving backwards from end to start of sequence. Connecting the two hidden states h_t and g_t calculates the final prediction of the Bidirectional RNN y_t.

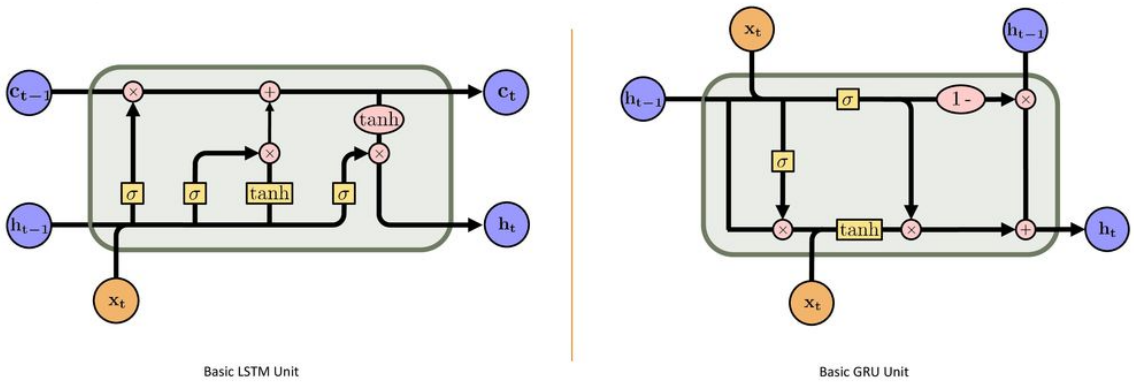


(Image Source)

FIGURE 4.6: Bidirectional RNNs

4.6.2 GRU

RNNs suffer from short-term memory, meaning that they have a hard time carrying information from earlier stages to later ones for longer sequences. This is due to the vanishing gradient problem during backpropagation, which means that the further the model progresses through the network, the lower the gradients get and subsequently it is harder to train the weights based on new information. One suggested approach for this issue suggested re-designing the simple RNN by adding special units that can hold information in memory longer. Thus, the LSTM and GRU architectures were proposed (Figure 4.7).



(Image Source)

FIGURE 4.7: RNN Units: LSTM vs GRU

Both of these networks were explicitly designed to avoid the long-term dependency problem. The LSTM was the first approach proposed in 1997 [16], followed by the GRU in 2014 [17]. Both architectures use gating mechanisms which are nothing but neural networks inside the LSTM and GRU cells, meaning that each gate has its own weights and biases. They are capable of learning which inputs in the sequence are important and store the information obtained by them in the memory unit, to be used by the following cell in the

sequence. The GRU model combines the "forget" and "input" gates of the LSTM into a single "update gate", making it in turn more efficient. Consequently, the GRU model is simpler than standard LSTM models, and can execute faster since it now uses less training parameters. GRU's are preferred when dealing with long sequences, as they take up less memory than the LSTM.

4.7 Hybrid Approach

Finally, a Convolutional-Recurrent Neural Network approach using a combination of the architectures of the best performing CNN and RNN model for Mel-Spectrograms is attempted. As mentioned above, a CNN can be very effective at automatically extracting and learning features both from two-dimensional image data as well as one-dimensional sequence data. Additionally, an RNN remembers every information through time and assumes a dependence between an input and its previous ones, and a bi-directional can also utilize future information. These two architectures are combined in a hybrid model where the CNN is regarded as the feature extractor and the RNN is utilized to learn the long-term patterns of the Mel-Spectrogram.

A typical song structure includes a verse, chorus, and bridge in the following arrangement: intro, verse — chorus — verse — chorus —bridge — chorus — outro. This is known as an ABABCB structure, where A is the verse, B is the chorus and C is the bridge[1]. The success of a song more often than not depends on the chorus, which means that the models are probable to focus mostly on the B parts. Additionally, as seen in Figure 3.1, loudness is positively correlated with popularity, which would imply that points of higher amplitude and their position in the Mel-Spectrogram could influence its success. Extracting features with CNNs and learning sequential information by a recurrent network can both help in emphasizing the importance of these points, and reducing the influence of irrelevant parts in the song.

4.8 Evaluation Measures

The measures that will be used for evaluating our models will be MAE, MSE and Spearman's Correlation.

Mean Absolute Error, or **MAE** for short, is the average absolute error between actual and predicted values. It basically computes, on average, the predicted value's distance from the true value (e.g if the true popularity score for a song is 60 and the predicted one is 70, the MAE for that song would be 10, which is equal to the distance between these scores).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| \tag{4.2}$$

where x_i and y_i are the observed and predicted values respectively and n is the number of samples.

[1] *How To Structure A Song?* by Carry A Tune Studio

In statistics, **MSE** or **Mean Squared Error** of an estimator measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is always strictly positive (it can zero only in ideal, unrealistic scenarios). The smaller the MSE is, the stronger the assumption that our model works well. One thing to be noted is that the MSE is sensitive to outliers. Even though our dataset is unbalanced, the highest popularity scores is not too distanced from the other scores, thus they are not considered outliers. The MSE can be a good indicator of how well a model fits the data, and can also be used for the comparison of potential models, by giving an indicator of choosing one model over another.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{4.3}$$

where x_i and y_i are the observed and predicted values respectively and n is the number of samples.

Usually, the thing that interests us the most is not just how popular a song will be, but also which song will be the most popular out of a collection of songs. For that reason, a new measure will also be taken into account. **Spearman's Rank Correlation Coefficient**, (ρ, also signified by rs) measures the strength and direction of association between two ranked variables. **SRCC** is simply a statistical test that assigns ranks to the value of each random variable and then computes Pearson Correlation Coefficient out of these ranks. In our case, the predicted scores are sorted from lower to higher and each song is assigned a rank, depending on its popularity prediction.. Then, the SRCC is computed by comparing the ranks occurring from the model predictions, and the ranked *y_test*.

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{4.4}$$

where d are the pairwise distances of the ranks of the variables x_i and y_i, and n is the number of samples.

Spearman's correlation coefficient ranges from -1 to +1. The sign of the coefficient indicates whether it is a positive or negative monotonic relationship between the ranked songs.

Finally, **R-squared** (R^2), also called the coefficient of determination is a statistical measure used in regression models that explains to what extent the variance of the dependent variable can be explained by the independent variables. In other words, R^2 shows how well the data fit the regression model (goodness of fit) and is calculated as:

$$R^2 = 1 - \frac{RSS}{TSS} \tag{4.5}$$

where are RSS is the Residual Sum of Square, and SST is the Total Sum of Squares.

R^2 values range from 0 to 1. In general, the higher the R^2, the better the model fits the data. This measure is not prioritised in model comparison and is only calculated as an additional indication of how well the model fits the data.

Chapter 5

Experiments

The experiments are separated in two categories: before the song is released and afterwards. This is due to the fact that the data concerning YouTube Views cannot be known before the release of the song and as a result will be excluded from the first phase of training. In both cases, the features will be approached separately, and then combined.

Hyperparameter tuning was only applied in some cases, due to lack of resources. In these cases, the hyperparameters were tuned for both tree based models[1] and neural networks[2]. The loss function used for the total of these models is Log-Cosh Loss[3], which is defined as the logarithm of the hyperbolic cosine of the prediction error. Essentially, it behaves similarly to the MSE for regression models, but additionally implements a degree of built-in protection against "wildly incorrect predictions" that are likely caused by outlier samples, which in our case are the most popular songs. All models were initially fitted to train for a total of 100 epochs (with the addition of callback methods), with 16 batch size (after many experiments, it was concluded that this batch size allows the model to learn better). Early stopping and reduction of learning rate are applied when the training seems to plateau. Additionally, the weights that produce the lowest validation loss during training are saved for every model, and all predictions are made using these weights.[4] All predicted popularity scores are rounded to the nearest integer.

5.1 Before Song Release

5.1.1 Spotify Features

Spotify Features are in tabular format. For that reason, too complex models are redundant; simple Machine Learning models are expected to perform better. The hyperparameters for these models were only hand-tuned, as these models were quite robust due to the simplicity of the dataset. A baseline regressor is also created for comparison, which always

[1]The implementation of Grid Search by sklearn was used. See https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html for more details.

[2]Bayesian Optimization was used from the KerasTuner package. See https://keras.io/keras_tuner/ for more details.

[3]Log-Cosh Loss Implementation: https://www.tensorflow.org/api_docs/python/tf/keras/losses/log_cosh

[4]All of the above actions are callback methods implemented from TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks

predicts the mean of the training set. Since Spotify features will also be used in combination to Mel-Spectrograms, whose processing requires a deep learning neural network, a simple MLP using Dense and Dropout layers is also implemented as to determine the best working hyperparameters for these features and additionally observe how they perform on their own.

Regression Approach

	MAE	**MSE**	**Spearman's Correlation**	**R^2**
Baseline	15	328.9	-0.05	0.0
Random Forest	14.631	**312.2**	0.222	**0.88**
XGBoost	14.67	313.1	0.223	0.25
LightGBM	**14.63**	313.4	**0.224**	0.31
(Tuned) MLP	14.79	317.6	0.186	0.03

TABLE 5.1: Performance of Spotify Features Before Song Release

Table 5.1 shows that the Random Forest and LightGBM models seem to have the best performance on Spotify Features, alas they are not much better than the other models. An average MAE of 14 in all models means that, on average, the popularity predicted differs 14 points from the actual popularity. This score could possibly indicate whether a song is too popular, or not at all. However, none of these models are able to significantly outperform the baseline operator, and as a result the conclusion can be reached that these features are not sufficient in popularity prediction.

5.1.2 Genres

After preprocessing, the genres dataset consists of four variables: genre(s), subgengre(s), the combination of the two and finally the first word of the genre(s). Since they are correlated, experiments are conducted with each feature separately in order to determine which format of the genres performs the best. Two quite different approaches were attempted:

- **Numerical:** Encoding each variable into numerical data by converting them into indicator variables.

- **Text:** Creating genre embeddings by approaching the genres as the resemblance of a sentence, which can also take into account the sequence of genres in each song.

The numerical approach, using only the first word of every genre and discarding the rest of the features, provides the best results for every model. This can to be explained due to the fact that more unique genres would produce more new features, which could make it difficult for the model to clearly differentiate between songs and could lead to overfitting, also known as the curse of dimensionality. The text attempts were not as fruitful, despite the fact that they are considered more advanced models. Doc2Vec and Sentence-Bert embeddings were created, but regardless of any dimensionality reduction techniques

attempted, they were not able to accurately represent all genre(s) and subgenre(s) of a song.

Regression Approach

	MAE	**MSE**	**Spearman's Correlation**	**R^2**
		Spotify Features		
LightGBM	**14.63**	313.4	**0.224**	0.31
		Genres		
Baseline	15	328.9	-0.05	0.0
Random Forest	14.139	304.07	0.2897	0.0822
XGBoost	**14.128**	304.69	**0.2899**	**0.0828**
LightGBM	14.131	**298.94**	0.2883	0.0657
(Tuned) MLP	14.447	308.37	0.2275	0.0622

TABLE 5.2: Performance of Genres Before Song Release

As seen on Table 5.2, song Genres seem to perform considerably better than the Spotify Features, with XGBoost reporting an MAE score of 14.1 and Spearman's correlation of 0.29. This is reasonable, as the songs that seem to dominate the charts are more often than not of a specific genre, most likely pop or hip-hop, in contrast to more acoustic or rock songs. Just the genre of the song is definitely not sufficient in predicting the popularity of a song, but could contribute significantly in combination with other features.

5.1.3 Mel-Spectrograms

A Mel-Spectrogram is a 128 dimensional matrix. As explained in Section 3.3.3, MFCCs were extracted from the Mel-Spectrograms, mostly for the cause of being used in tabular format. The same models as above are applied for the MFCCs dataset that also includes the delta coefficients, with hand-tuned hyperparameters. As far as the Mel-Spectrograms are concerned, they are approached in two different directions:

- **Image:** A Mel-Spectrogram is essentially an 128x1292 image. As indicated by many applications in image deep learning, but also suggestions from previous researches, a 2-D Convolution model is implemented for prediction. The goal of the Convolution layers is essentially to detect features in the spectrogram, which could be interpreted as the parts of a song contributing the most in its popularity. In addition, the pre-trained models VGG-16 and DenseNet are also used and transfer-learning is applied to influence the model weights for this specific task.

- **Sequences:** In Mel-Spectrograms, the two dimensions represent fundamentally different units, one being strength of frequency and the other being time. Due to the

one representing time, a sequential approach is attempted using various combinations of 1-D Convolution and Bi-Directional GRU layers. Considering that a Mel-Spectrogram is divided into 1292 time segments of a song, one sequence is basically a 128-D vector. Consequently, a 1292x128 array is given as input to these models.

The Neural Network models for Mel-Spectrograms were not tuned due to lack of resources, as spectrograms are quite large files and their processing requires a lot of computational power. As a result, some hyperparameters were hand-picked after running the models for a few times. Of all combinations of layers attempted, the best performing models and combinations of them are presented Table 5.3:

Regression Approach

	MAE	**MSE**	**Spearman's Correlation**	**R^2**
		Spotify Features		
LightGBM	**14.63**	313.4	**0.224**	0.31
		Genres		
XGBoost	**14.12**	304.69	**0.289**	**0.08**
		MFCCs		
Baseline	15	328.9	-0.05	0.0
Random Forest	**14.7**	**319.8**	**0.156**	0.86
XGBoost	15.1	333.1	0.132	0.57
LightGBM	14.8	321.1	0.144	0.18
		Mel-Spectrograms		
Image Approach				
2D CNN	14.8	326.1	0.122	0.01
2D CNN + Bi-GRU	14.9	324.5	0.140	0.01
VGG16	15.1	348.2	0.175	-0.03
VGG16 + Transfer Learning	15.1	350.0	0.176	-0.04
DenseNet	14.9	337.3	0.145	-0.01
DenseNet + Transfer Learning	14.9	336.0	0.153	-0.00
Sequence Approach				
1D CNN	**14.79**	325.0	**0.187**	0.01
1D CNN + Bi-GRU	14.80	**323.8**	0.184	0.01

TABLE 5.3: Performance of Mel-Spectrograms Before Song Release

The sequence approach has great performance on popularity prediction using just the Mel-Spectrograms; even though Spotify features and genres have better performance, the 1-D CNN performs adequately with MAE equal to 14.79 and Spearman's Correlation of 0.187. Once again, the tree-based models prove their stability when trained on the MFCCs,

coming very close to the scores of the more complex neural networks and even surpassing some of the 2-D CNN models. Surprisingly, the VGG16 pre-trained model achieves the highest Spearman's Correlation score amongst the 2-D CNN models. Transfer learning on both pre-trained models doesn't seem to influence the performance of the final model.

5.1.4 Spotify Features + Genres + Mel-Spectrograms

In the tabular approach, a concatenation of all features in a single dataframe is given as input to the models. In this case, all models are tuned to account for the variability of the different inputs. In the deep learning approach, the tuned MLP model versions for audio and genres are used, in addition to the two best performing models for Mel-Spectrograms for both image and sequence approaches. This is done to observe whether the interactions with the new data could improve the final result. The models expect three different inputs, which are passed through layers on their own and are later concatenated for the final prediction.

Another approach worth mentioning is the embedding similarity search. An embedding is a relatively low-dimensional space into which one can translate high-dimensional vectors[5]. According to that method, embeddings are extracted for every song using the best performing neural network, each containing information regarding all features known before the song release: Spotify tags, genres and the Mel-Spectrogram. Every song has now a one dimensional vector representation. The new song's popularity is assigned the mean score of the most similar song embeddings, found by computing the cosine similarity between the song vectors. Then, the most similar songs to a new song are detected, by computing the cosine similarity between the two vectors.[6] Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors x and y and determines whether they are pointing in roughly the same direction[7]. It is calculated as:

$$Cosine(x, y) = \frac{x \cdot y}{\|x\|\|y\|} \tag{5.1}$$

The popularity score is then calculated by taking the mean popularity scores amongst these songs. The number of closest songs used to compute the score of a new song was hand-tuned and was concluded to be 40.

[5]Embeddings by Google Developers: https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture

[6]The implementation of the Nearest Neighbours algorithm by sklearn was implemented. See https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html for more details.

[7]Cosine Similarity by Engati: https://www.engati.com/glossary/cosine-similarity

Spotify Features + Genres + Mel-Spectrograms **Regression Approach**

	MAE	MSE	Spearman's Correlation	R^2
with MFCCs				
Baseline	15	328.9	-0.05	0.0
(Tuned) Random Forest	14.2	297.0	0.319	0.7
(Tuned) XGBoost	14.13	291.9	0.324	0.3
(Tuned) LightGBM	**13.89**	288.1	**0.343**	0.2
with Mel-Spectrograms				
Image Approach				
(Tuned) MLP + 2D CNN	15.19	334.3	0.02	0.01
(Tuned) MLP + VGG16	15.17	333.5	0.14	0.01
Sequence Approach				
(Tuned) MLP + 1D CNN	**13.68**	284.2	**0.365**	0.1
(Tuned) MLP + 1D CNN Embeddings + Cosine Similarity	14.09	295.2	0.31	-

TABLE 5.4: Performance of Combined Features Before Song Release

Here, immediate comparison between tree models and more complex deep learning networks can be applied. Surprisingly, the tree models outperform most of the neural networks, with LightGBM coming first with a MAE of 13.89, MSE of 288.17 and Spearman's Correlation of 0.34. This is not so unexpected if the size of the dataset is taken into consideration as just 8,000 songs are used for training. Similarly, the simplest MLP also seemed to struggle when trained on Spotify Tags and Genres. Despite that, considerable performance is also achieved by the hybrid model using a MLP architecture with tuned hyperparameters for Spotify Features and Genres and 1-D CNN architecture for Mel-Spectrograms, producing the highest Spearman's Correlation of 0.365.[8]

Figure 5.1 shows how challenging it is to detect hit songs. Ideally, all points in this visualization would form a diagonal, indicating that the predicted popularity score and the golden truth coincide. Almost all songs are predicted in the low popularity range, and no songs with a popularity over 40 are predicted correctly.

5.1.5 Feature Importance

When using too many features, each coming from datasets quite different in nature, it would be interesting to see which of these contributes the most when they are trained together. Scott Lundberg and Su-In Lee in 2017 introduced a new method for interpreting predictions and feature importance using the unified framework **SHAP** [28]. The calculation of SHAP (SHapley Additive exPlanations) values is a method based on cooperative

[8]Due to the large size of this network, please refer to Appendix B for the full model architecture.

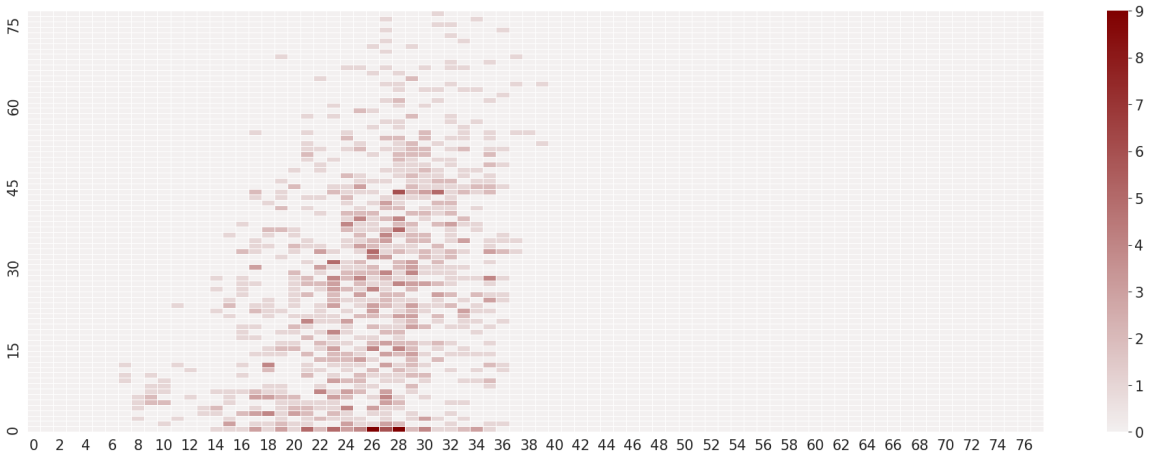


FIGURE 5.1: Distribution of LightGBM Predictions (Y Axis) vs Actual Values (X Axis) Before Song Release.

game theory and is used to increase transparency and interpretability of machine learning models. A SHAP value basically quantifies, on average, the magnitude (positive or negative) of each feature's contribution towards the predicted popularity. Features with higher mean absolute SHAP values are more influential.[9]

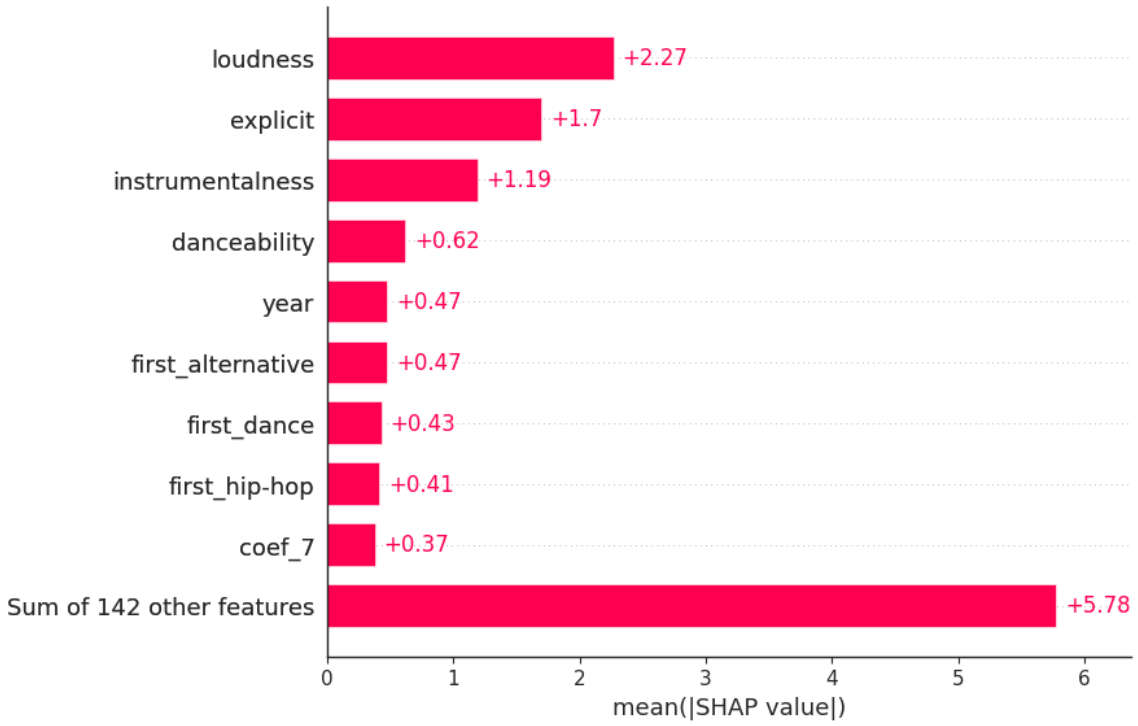


FIGURE 5.2: Shap Values of LightGBM Predictions Before Song Release.

Previous experiments showed that Genres seemed to have the biggest prediction strength for popularity, with Spotify Features coming second and Mel-Spectrograms last. In Figure 5.2, we can see exactly which features contribute the most when they are combined

[9]For the computation of these values, the SHAP package is used: https://shap.readthedocs.io/en/latest/index.html

together, on the best performing machine learning model LightGBM. The 10 most influential features here are ordered from the highest to the lowest effect on the song popularity. The absolute SHAP value is used, so it does not matter if the feature affects the popularity in a positive or negative way. It seems that *loudness* is the most influential variable, contributing on average ± 2.27 to each song's predicted popularity, followed by whether the song has *explicit* lyrics or not and its *instrumentalness*. The added feature of the *year* the song was released also seems to influence a song's predicted popularity, as well as the genres *alternative*, *dance* and *hip-hop*. As far as high level audio features are concerned, only the mean of the 7th MFCC as indicated by the feature *coef_7* seems to have a significant contribution to the final prediction.

5.2 After Song Release

One might question what the purpose of the YouTube Views is, as using them for training would imply that a song has already been released so subsequently its popularity is already known. Indeed, this is a dilemma that was often raised during the writing of this thesis. However, there are many cases in which the industry is interested in the further evolvement of an already released song's popularity. Let's assume that a song has been out for a while. This song could either be just a bit old (a month), or quite older (more than a year). In the first case, there is much interest in how much budget will be allocated in an already released song, judging by how it has performed so far. For instance, if its predicted popularity is found to be low, that would imply that this song was not well received and as a result its funding might be reduced, or different ideas might be considered for its promotion. The views provided are monthly, concerning a total of 41 months, but in a similar manner daily views concerning just one month could also be used. In the second case, a new phenomenon of many old songs resurfacing in popularity has been observed, due to new social media platforms focusing on music-related entertainment content such as TikTok. Additionally, many older albums or new album compilations using old songs are also being released, either for legal reasons between recording labels or as anniversaries/celebrations of the artists contribution to the music industry. In both cases, the information of how this song performed in the past could be proven beneficial in predicting how its popularity might increase now, and could influence the promotion of an older song.

5.2.1 YouTube Views

YouTube Views are essentially time-series data. As previously, they will also be approached as tabular data, using the extracted static features on models like Random Forest and XGBoost, as well as time-series data as input to neural networks specifically designed for sequential data, such as 1D Convolution and Bidirectional-GRU. The neural networks take as input for each song an array of 2 features and 41 timesteps, for a task of multivariate time-series regression. This time they were initially fitted for a total of 200 epochs, as information learning seemed to increase the more the model was trained and callbacks did not stop the model training in the first 100 epochs. The best performing models are presented in Table 5.5:

Regression Approach

	MAE	**MSE**	**Spearman's Correlation**	**R^2**
Spotify Features				
LightGBM	**14.63**	313.4	**0.224**	0.31
Genres				
XGBoost	**14.128**	304.69	**0.2899**	**0.0828**
MFCCs				

	MAE	MSE	Spearman's Correlation	R^2
Random Forest	**14.7**	**319.8**	**0.156**	0.86
Mel-Spectrograms				
1D CNN	**14.79**	325.0	**0.187**	0.01
Static Features				
Baseline	15	328.9	-0.05	0.0
Random Forest	**9.80**	**171.53**	0.674	0.94
XGBoost	9.86	174.61	0.671	0.71
LightGBM	9.82	172.05	**0.676**	0.68
Time Series				
(Tuned) 1D CNN	10.89	244.25	0.66	0.25
(Tuned) Bi-GRU	10.19	**185.03**	**0.661**	0.43

TABLE 5.5: Performance of YouTube Features After Song Release

As Table 5.5 indicates, YouTube Views boost significantly the model performances. The extracted static features seem to work just as well as the raw time-series, even slightly better, as the simplest Random Forest has the smallest MAE of 9.80 and LightGBM the highest Spearman's correlation of 0.676. This once again confirms the robustness of the tree-based models.

5.2.2 Spotify Features + Genres + Mel-Spectrograms + YouTube Views

Finally, all features are now concatenated and trained. Only 1-D Convulotional networks were used for the Mel-Spectrograms, as they seemed to perform the best. The hybrid model using a MLP architecture with tuned hyperparameters for Spotify Features and Genres, 1-D CNN architecture for Mel-Spectrograms and a Bidirectional GRU architecture with tuned hyperparameters for YouTube Views is also used for the creation of unique embeddings for every song. As previously, the popularity score for each song is calculated according to the popularity scores of the most similar songs, found using cosine similarity. This time, the number of closest songs used to compute the score of a new song is 35.[10]

Spotify Features + Genres + Mel-Spectrograms + YouTube Views **Regression Approach**

	MAE	MSE	Spearman's Correlation	R^2
MFCCs + Static Features				
Baseline	15	328.9	-0.05	0.0
(Tuned) Random Forest	9.5250	162.06	0.69742	0.9
(Tuned) XGBoost	**9.1970**	**153.947**	**0.71292**	0.7

[10] Due to the large size of this network, please refer to Appendix B for the full model architecture.

	MAE	MSE	Spearman's Correlation	R^2
(Tuned) LightGBM	9.1105	156.44	0.71005	0.7
Mel-Spectrograms + Time Series				
Sequence Approach				
(Tuned) MLP + 1D CNN + (Tuned) Bi-GRU	9.25	157.73	**0.7135**	0.5203
(Tuned) MLP + 1D CNN + (Tuned) Bi-GRU + Embeddings + Cosine Similarity	9.86	172.95	0.704	-

TABLE 5.6: Performance of Combined Features After Song Release

Once again, the tuned tree-based models perform greatly, with XGBoost achieving a MAE of 9.19 and Spearman's Correlation of 0.712. However, the best Spearman's correlation of 0.7135 is now achieved by the hybrid deep learning model.

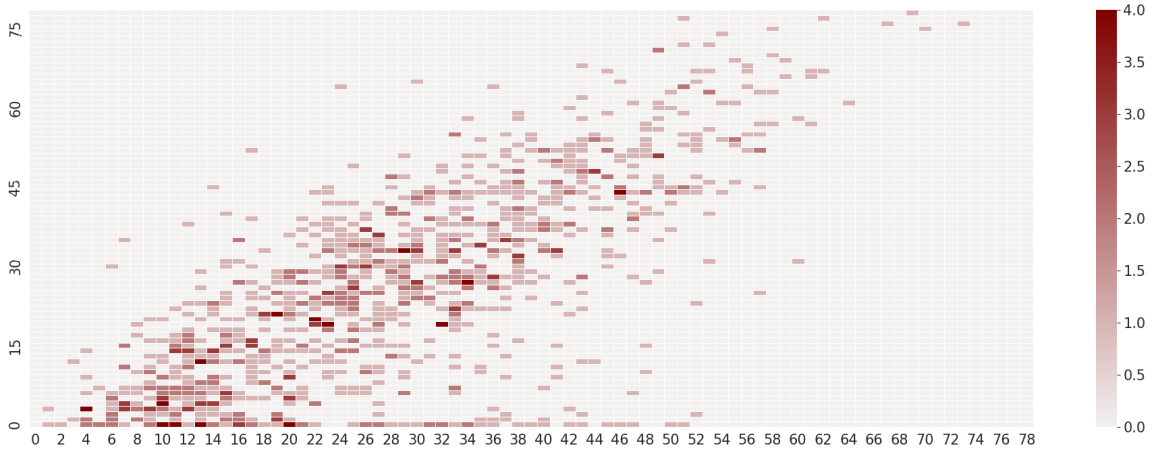


FIGURE 5.3: Distribution of XGBoost Predictions (Y Axis) vs Actual Values (X Axis) After Song Release.

This time, prediction and golden truth values are much closer to the diagonal of the visualization in Figure 5.3. Hit songs still struggle at getting detected, however now popularity scores over 40 are more likely to be predicted correctly.

5.2.3 Feature Importance

As seen on all of the experiments above, the inclusion of YouTube Views in training offers a significant performance boost for all models. This comes to no surprise, since these features already have some information about the success of a song. For that reason it would be interesting to examine whether the impact of features concerning YouTube Views dominates when combined with the rest of the features, as would be expected, or if any of the

other features also contribute in popularity prediction. Once again, the SHAP values are calculated and the results are presented in Figure 5.4.

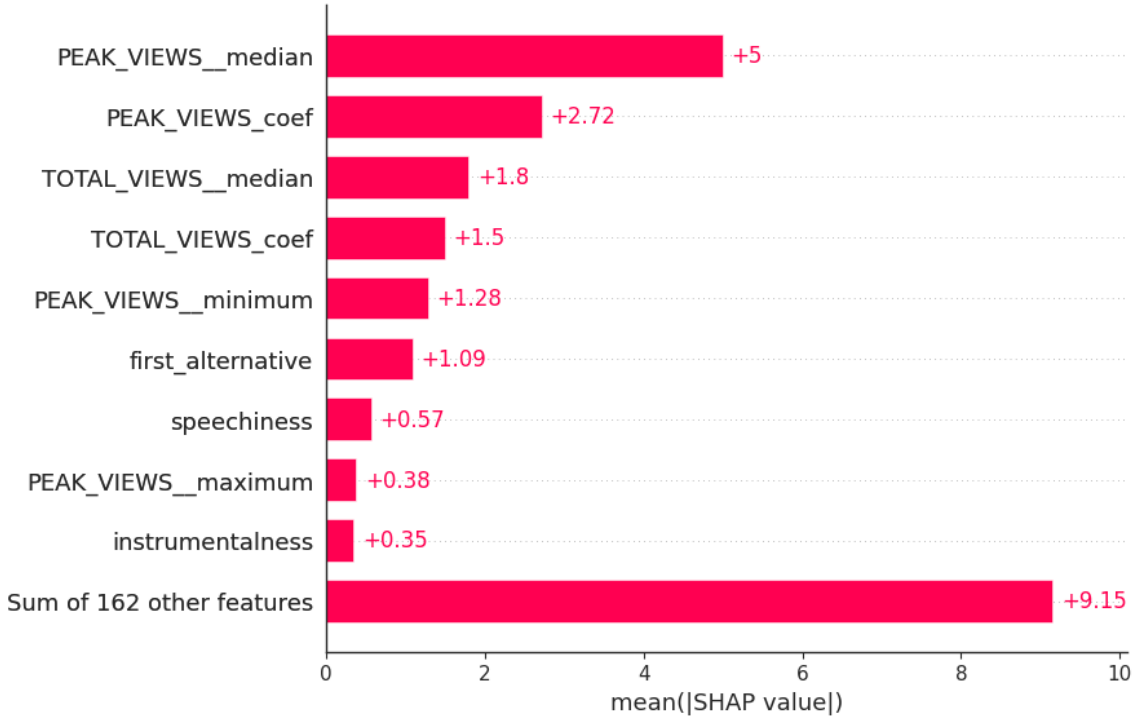


FIGURE 5.4: Shap Values of XGBoost Predictions After Song Release.

As expected, features concerning YouTube Views strongly influence the popularity score, with the *median* of Peak Views contributing to the song's popularity score up to 5 points. However, the *alternative* genre as well as the *speechiness* and *instrumentalness* of the song also seem to have some influence on the final popularity score.

As an end to the Experiments part of this thesis, it can be concluded that a regression approach to popularity prediction can give us a good estimation of whether a song will be successful or not, but detecting hit songs specifically is a more tricky task. This is why a classification approach is also attempted, using the best performing models from above as they seem to be able to learn the data most efficiently, now in a classification scenario. Each song now has a new categorical label indicating its popularity: lower, medium and higher. The aim of this approach is to examine how the models above perform in the higher popularity class.[11]

[11] For further information on this approach, please refer to Appendix A.

Chapter 6

Conclusions

In this project we dealt with four different types of data concerning songs released from the start of 2017 to the end of 2018 for the aim of predicting the song's popularity and ultimately detecting hit songs. These data are separated in two different training phases; the first one only utilizes data that can be known prior to the release of the song and the other takes into consideration additional features known after the release of the song. The first training phase included low-level audio features such as Spotify Tags, Genres and Subgenres, high-level features such as Mel-Spectrograms and the Mel-frequency cepstral coefficients (MFCCs) extracted by them. The second phase of training uses all features above with the addition of monthly YouTube Views concerning approximately the past 4 years, as they are only known after the song is released.

Multiple approaches were attempted, utilizing the datasets both in strictly tabular format as input to tree-based models, as well as their raw format as input to neural networks. In the first training phase, which includes all data known before the song release, the best performing models were LightGBM, achieving a MAE of 13.89, MSE of 288.1 and Spearman's Correlation of 0.343, and the hybrid MLP-CNN model achieving 13.68 MAE, 284.2 MSE and 0.365 Spearman's Correlation. As far as modeling concerning data known after the release of the song, the addition of YouTube Views gives a performance boost of 9.1970 MAE, 153.947 MSE and 0.712 Spearman's Correlation on the best tree-based performing model XGBoost. The hybrid MLP-CNN-Bi-GRU neural network also achieved considerable performance with Spearman's Correlation of 0.7135, indicating that the ranking of the predicted score popularity very closely correlates their actual ranking. Comparatively, features from the YouTube Views dataset contributed the most to popularity prediction, as expected, followed by Spotify Tags and Genres. Choosing between these best-performing models depends on what is as most significant: the exact popularity score of the song, in which case MAE and MSE will be more heavily weighted, or the ranking of specific songs from least to most popular, in which case Spearman's correlation outweighs the rest of the metrics.

6.1 Problems Encountered

The biggest challenge faced in this thesis was the imbalanced nature of the dataset. Indeed, the best performing model using data known only before the song's release fails to detect any hit songs, even though they are present on the test dataset. Being able to detect a hit out of a big variety of songs where most are unlikely to succeed is a demanding task and a much more significant one than predicting if a song will be of low or medium popularity. For example, if a label would like to increase profits, they may choose to invest their limited resource for promotion on tracks that are likely to become popular. For that reason, the detection of hit songs is given additional weight as the number of hit songs used for training was quite small, only occupying around 2% of the total dataset, making their detection a challenging task.

6.2 Future Work

The current thesis mostly focuses on predicting the popularity of a song solely depending on its high and low level audio characteristics. The poor results of the models before YouTube Views are included for training are reasonable, as YouTube views are the greatest indicators to a song's success amongst all features. Moreover, there are many external factors that were not considered for this task, which could contribute just as much, if not more, to predicting a song's success.

The inclusion of additional features that are also known before the song release would definitely boost model performance. Indeed, knowing the song lyrics and the language they are written in (e.g. romance themed songs with English lyrics usually tend to be more relatable and get higher spots on the charts in comparison to metal songs in German) or who the artist(s) of the song is/are and information about them (e.g., their monthly listeners on Spotify up to now or whether they have been on the charts before) could significantly influence the model results. Additional features such as the songwriter, composer and producer of a song as well as the music label, what platforms it is being released on and when or specific marketing strategies could also be proved to be beneficial. We have to acknowledge that the models created naturally do not include any external factors such as the ones mentioned above, as the aim of the project was to focus on specific types of data. However, in industry problems, all these extrernal factors are usually known, accounted for and used in training, and could definitely influence the models towards the right direction.

Bibliography

[1] Mohamed Nasreldin et al. *Song Popularity Predictor*. May 2018. URL: https://towardsdatascience.com/song-popularity-predictor-1ef69735e380.

[2] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282.

[3] Naomi S Altman. "An introduction to kernel and nearest-neighbor nonparametric regression". In: *The American Statistician* 46.3 (1992), pp. 175–185.

[4] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. URL: http://doi.acm.org/10.1145/2939672.2939785.

[5] Kai Middlebrook and Kian Sheik. *Song Hit Prediction: Predicting Billboard Hits Using Spotify Data*. Aug. 2019. URL: https://arxiv.org/pdf/1908.08609.pdf.

[6] S. S. Stevens et al. *A scale for the measurement of the psychological magnitude pitch*. Acoustical Society of AmericaASA, Jan. 1970. URL: https://asa.scitation.org/doi/10.1121/1.1915893.

[7] Eva Zangerle et al. *Hit Song Prediction: Leveraging low- and high-level Audio Features*. Nov. 2019. URL: https://archives.ismir.net/ismir2019/paper/000037.pdf.

[8] David Martín-Gutiérrez et al. *A Multimodal End-to-End Deep Learning Architecture for Music Popularity Prediction*. Feb. 2020. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9007339.

[9] Kamalesh Palanisamy et al. *Rethinking CNN Models for Audio Classification*. Nov. 2020. URL: https://arxiv.org/pdf/2007.11154.pdf.

[10] Gao Huang et al. "Densely Connected Convolutional Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[11] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv preprint arXiv:1512.03385* (2015).

[12] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[13] Eleni Tsalera et al. *Comparison of Pre-Trained CNNs for Audio Classification Using Transfer Learning*. Oct. 2021. URL: https://www.mdpi.com/2224-2708/10/4/72/htm.

[14] Gopal Chandra Jana et al. *A 1D-CNN-Spectrogram Based Approach for Seizure Detection from EEG Signal*. 2019. URL: https://www.sciencedirect.com/science/book/pii/S1877050920307146?ref=pdf_download&fr=RR-2&rr=76aaca5afe916f3b.

[15] Iurii Lezhenin et al. *Urban Sound Classification using Long Short-Term Memory Neural Network*. Oct. 2019. URL: https://annals-csis.org/Volume_18/drp/pdf/185.pdf.

[16] Sepp Hochreiter and Jurgen Schmidhuber. *LONG SHORT-TERM MEMORY*. 1997. URL: http://www.bioinf.jku.at/publications/older/2604.pdf.

[17] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*. Sept. 2014. URL: https://arxiv.org/pdf/1406.1078v3.pdf.

[18] Zain Nasrullah and Yue Zhao. *Music Artist Classification with Convolutional Recurrent Neural Networks*. Mar. 2019. URL: https://arxiv.org/pdf/1901.04555.pdf.

[19] Nicholas Indorf. *Using Deep Learning to Predict Hip-Hop Popularity on Spotify*. Jan. 2022. URL: https://towardsdatascience.com/using-deep-learning-to-predict-hip-hop-popularity-on-spotify-1125dc734ac2.

[20] Yong Xu et al. *Convolutional Gated Recurrent Neural Network Incorporating Spatial Features for Audio Tagging*. Feb. 2017. URL: https://arxiv.org/pdf/1702.07787.pdf.

[21] Maria Habib et al. *Toward an Automatic Quality Assessment of Voice-Based Telemedicine Consultations: A Deep Learning Approach*. May 2021. URL: https://www.researchgate.net/publication/351469852_Toward_an_Automatic_Quality_Assessment_of_Voice-Based_Telemedicine_Consultations_A_Deep_Learning_Approach#pf1a.

[22] K Khadar Nawas et al. *Speaker Recognition using Random Forest*. 2021. URL: https://www.itm-conferences.org/articles/itmconf/abs/2021/02/itmconf_icitsd2021_01022/itmconf_icitsd2021_01022.html.

[23] *Mel Frequency Cepstral Coefficient (MFCC) tutorial*. URL: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

[24] Marshall Scott Poole et al. *Hypothesis testing and modeling perspectives on inquiry.* Handbook of interpersonal communication. SAGE, 2002, pp. 23–72.

[25] Xindong Wu et al. "Top 10 algorithms in data mining". In: *Knowledge and information systems* 14.1 (2008), pp. 1–37.

[26] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017), pp. 3146–3154.

[27] Karen Simonyan and Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition*. Apr. 2015. URL: https://arxiv.org/abs/1409.1556.

[28] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions*. Nov. 2017. URL: https://arxiv.org/abs/1705.07874.

Appendix A

Classification Approach

Imbalanced datasets are a common problem, especially in datasets concerning song popularity. As mentioned in Chapter 5, predicting the actual popularity score but my not be too singificant than predicting whether the song is a hit or not. For that reason, a new label is now added to every song, according to the rule:

- **[0, 29]:** If the song popularity score is in that range, it is classified as of low popularirty. 59% of the songs belong in this category.
- **[30, 58]:** If the song popularity score is in that range, it is classified as of medium popularirty. 39% of the songs belong in this category.
- **[59, 89]:** If the song popularity score is in that range, it is classified as of high popularirty. 2% of the songs belong in this category.[1]

A.1 Evaluation Metrics

The most commonly used metrics for classification are accuracy, precision, recall and f1-score. Since the dataset is imablanced, accuracy is not a reliable measure. The decision whether precision or recall will be preferred depends on whether the record label would rather falsely promote a not so promising song, in which case precision is preferred, or miss out on promoting a promising song that got classified as medium. F1 Score will be the most important metric as it combines both precision and recallinto a single metric by taking their harmonic mean. Due to the imbalanced nature of the dataset, only the weighted average (WA) of all metrics will be considered, where the weights assigned to each class are inversely proportional to their frequency in the dataset. Low WA precision, recall and f1-score would mean that the majority class is prefered in prediction.

For prediction, tree-based models are implemented as they proved to be quite robust and performed just as well as more complex and resource-consuming deep learning networks. KNN is also added, as it also takes into account the cosine distance between each song's features.

[1]The range is decided according to the pandas cut function: https://pandas.pydata.org/docs/reference/api/pandas.cut.html

A.2 Experiments

Classification Approach

	WA Precision	WA Recall	WA F-1 Score
Before Song Release			
Baseline	0.34	0.59	0.43
KNN	0.53	0.53	0.53
(Tuned) Random Forest	0.53	0.58	0.54
(Tuned) XGBoost	0.54	0.58	0.55
(Tuned) LightGBM	0.55	0.58	**0.56**
After Song Release			
Baseline	0.34	0.59	0.43
KNN	0.58	0.59	0.58
(Tuned) Random Forest	0.75	0.74	**0.74**
(Tuned) XGBoost	**0.76**	0.74	**0.74**
(Tuned) LightGBM	0.75	0.73	0.73

TABLE A.1: Performance of all Features (Classification)

Below are presented the confusion matrixes for the best performing models:

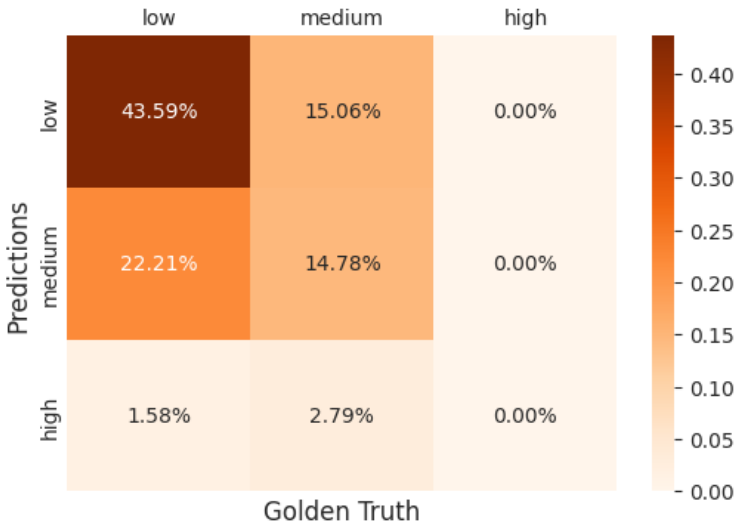


FIGURE A.1: Confusion Matrix of LighGBM Predictions Before Song Release.

Using only the data known before the song release for training is not, a 54% Weighted Average F1 Score. However, as seen on A.1 the best performing model fails to predict any song on the higher popularity category.

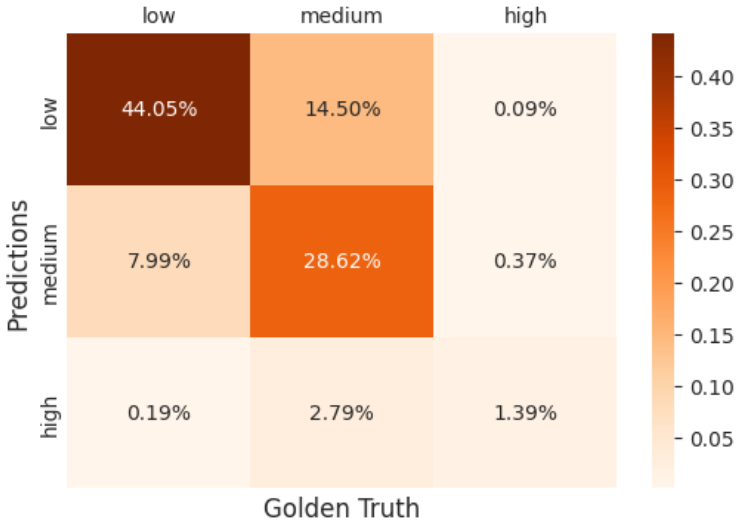


FIGURE A.2: Confusion Matrix of XGBoost Predictions After Song Release.

Having more information about the song (YouTube Views) increases the Weighted Average F1 Score to 74%; more medium popularity a lot more songs are now classified correctly and hit songs are not being completely ignores as a few songs are now being categorized as high popularity overall.

Appendix B

Model Architectures

The architectures of the best performing neural networks for hit song prediction before and after the song release are presented below.

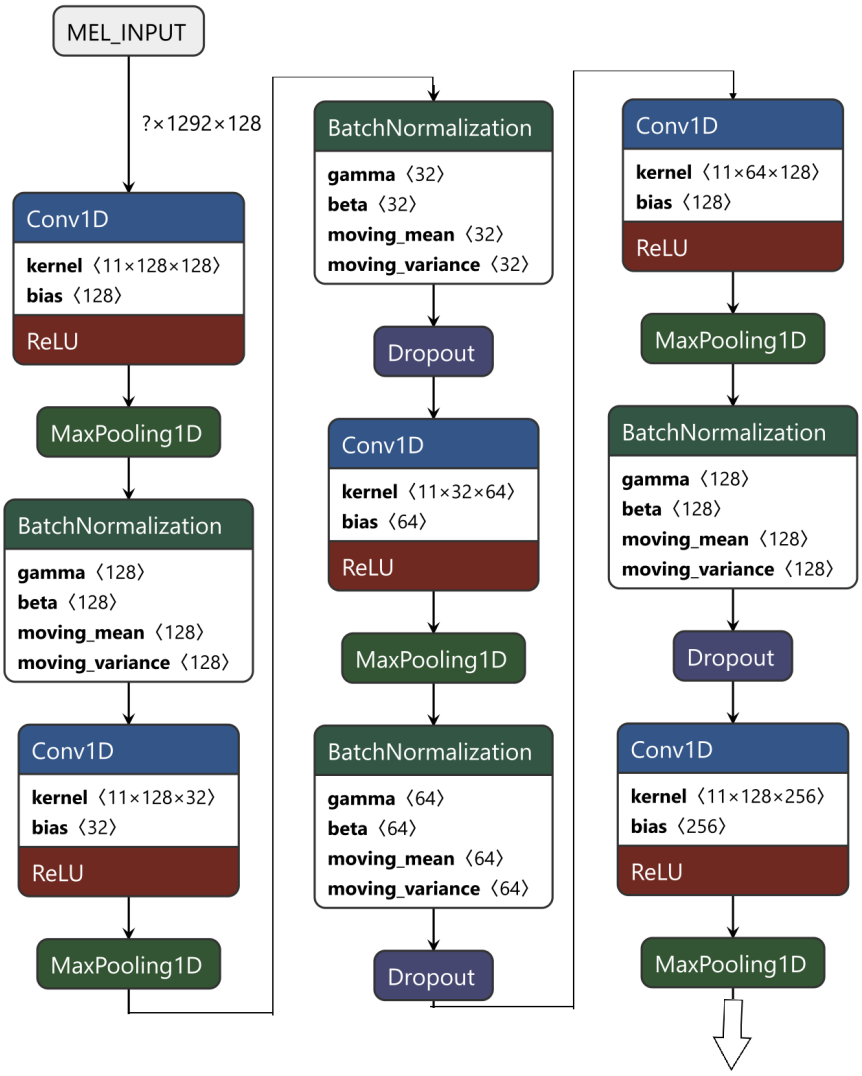


FIGURE B.1: Architecture of Hybrid Model for popularity prediction Before Song Release (Part 1 of 3).

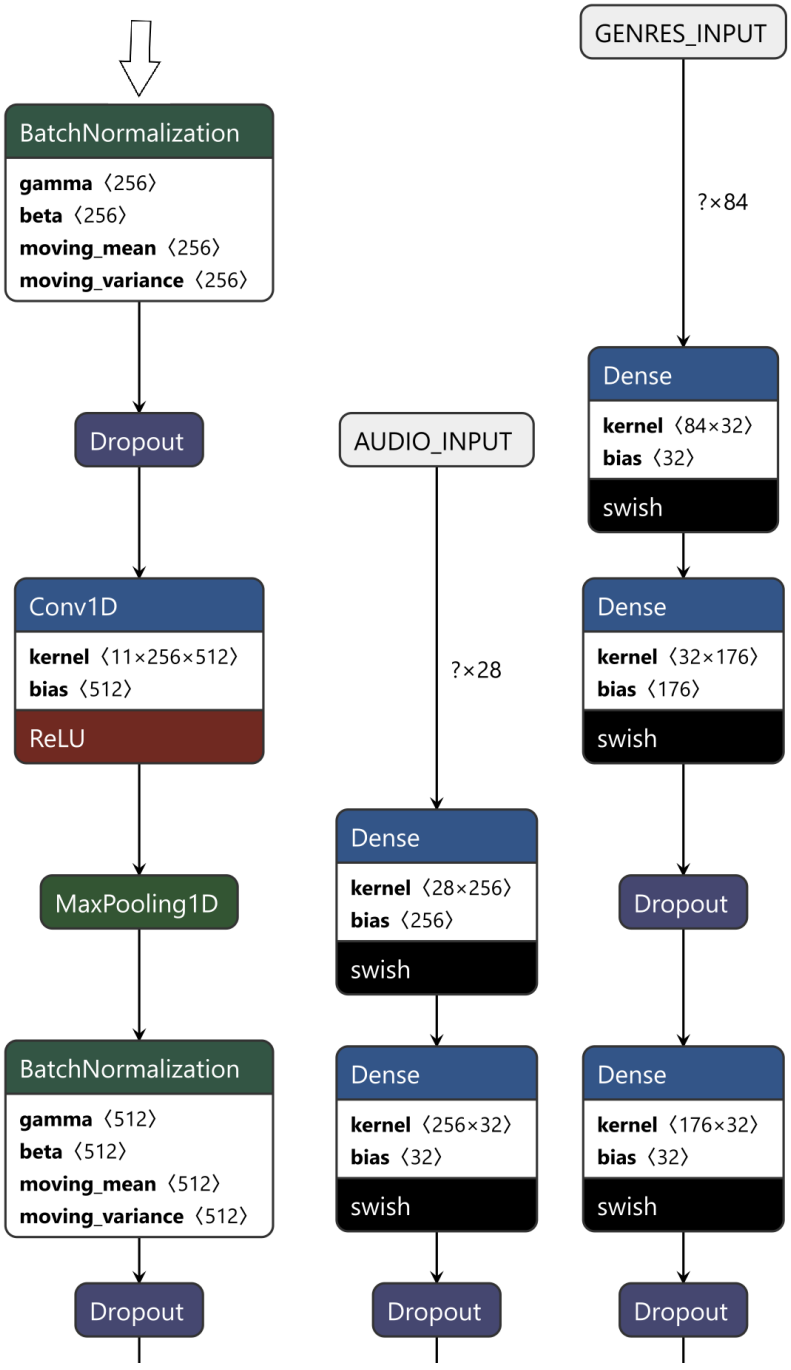


FIGURE B.2: Architecture of Hybrid Model for popularity prediction Before Song Release (Part 2 of 3).

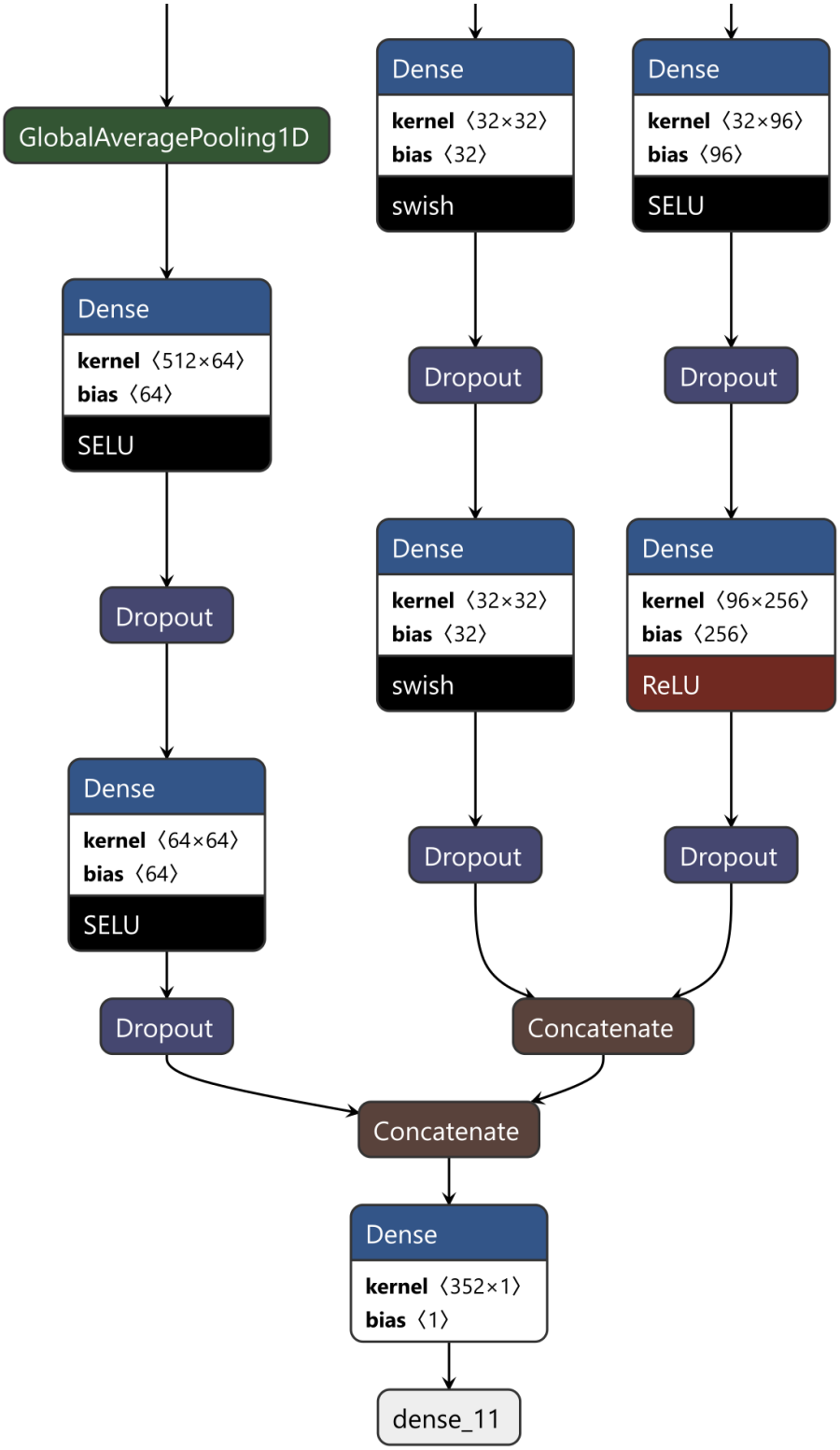


FIGURE B.3: Architecture of Hybrid Model for popularity prediction Before Song Release (Part 3 of 3)

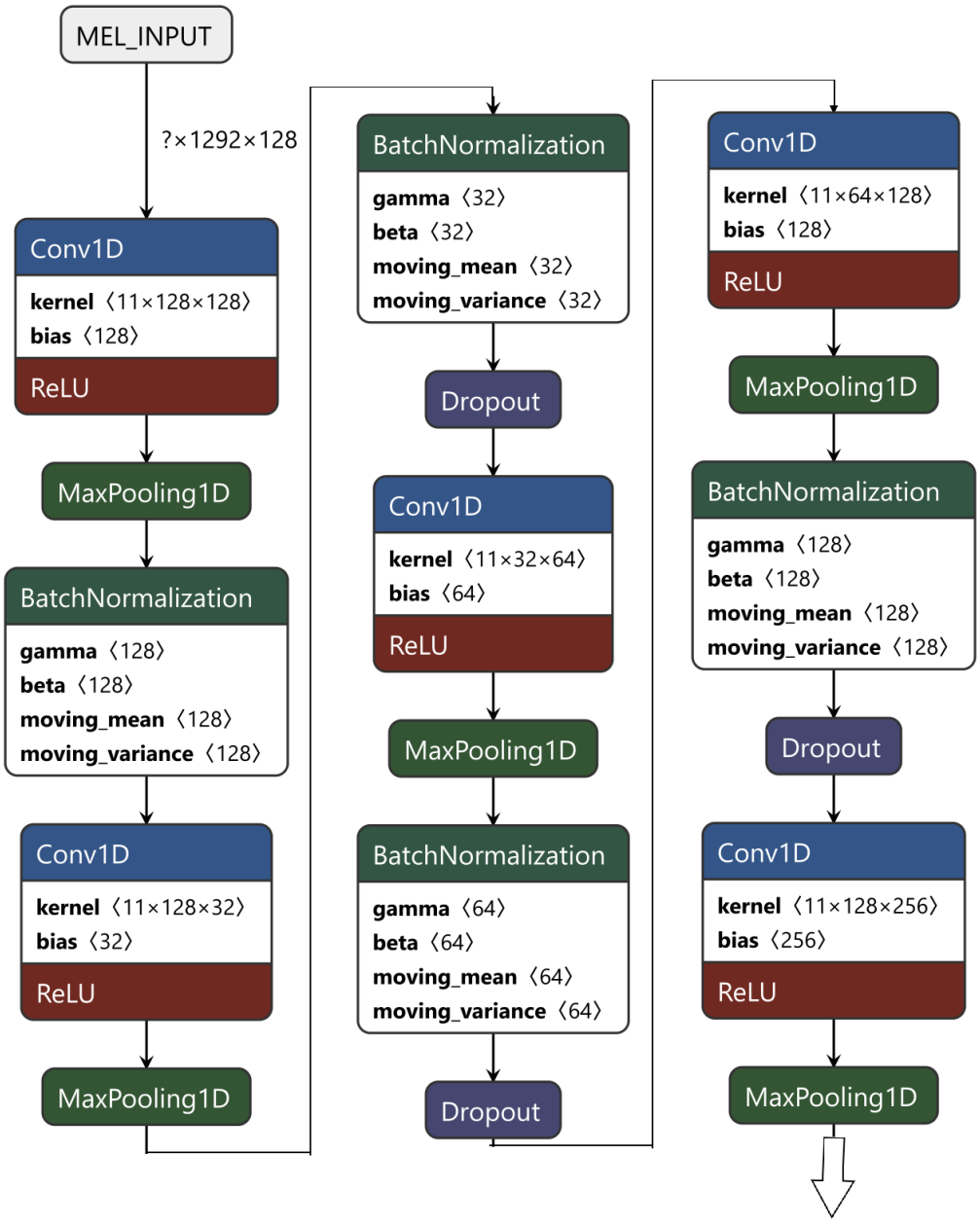


FIGURE B.4: Architecture of Hybrid Model for popularity prediction After Song Release (Part 1 of 3).

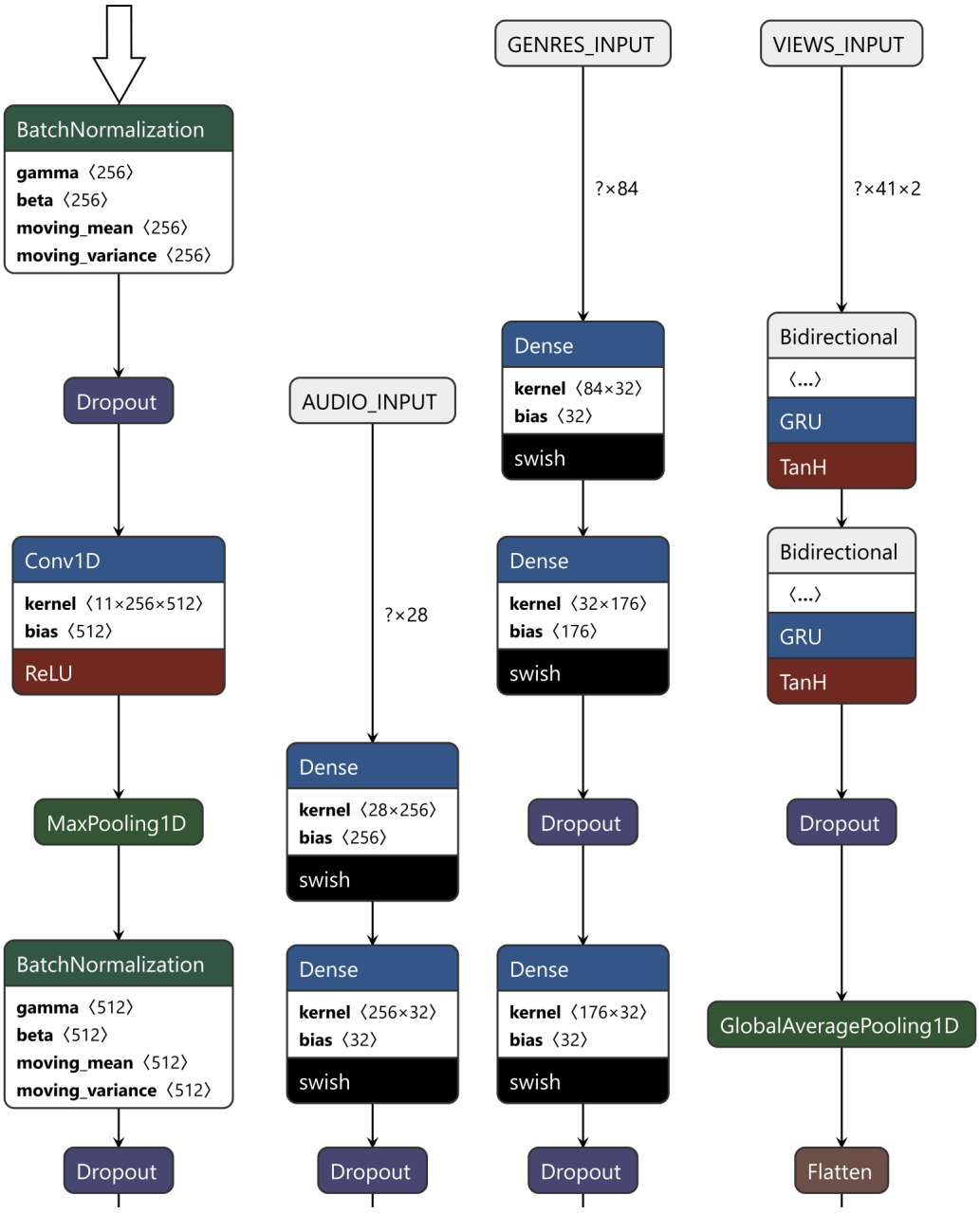


FIGURE B.5: Architecture of Hybrid Model for popularity prediction After Song Release (Part 2 of 3)

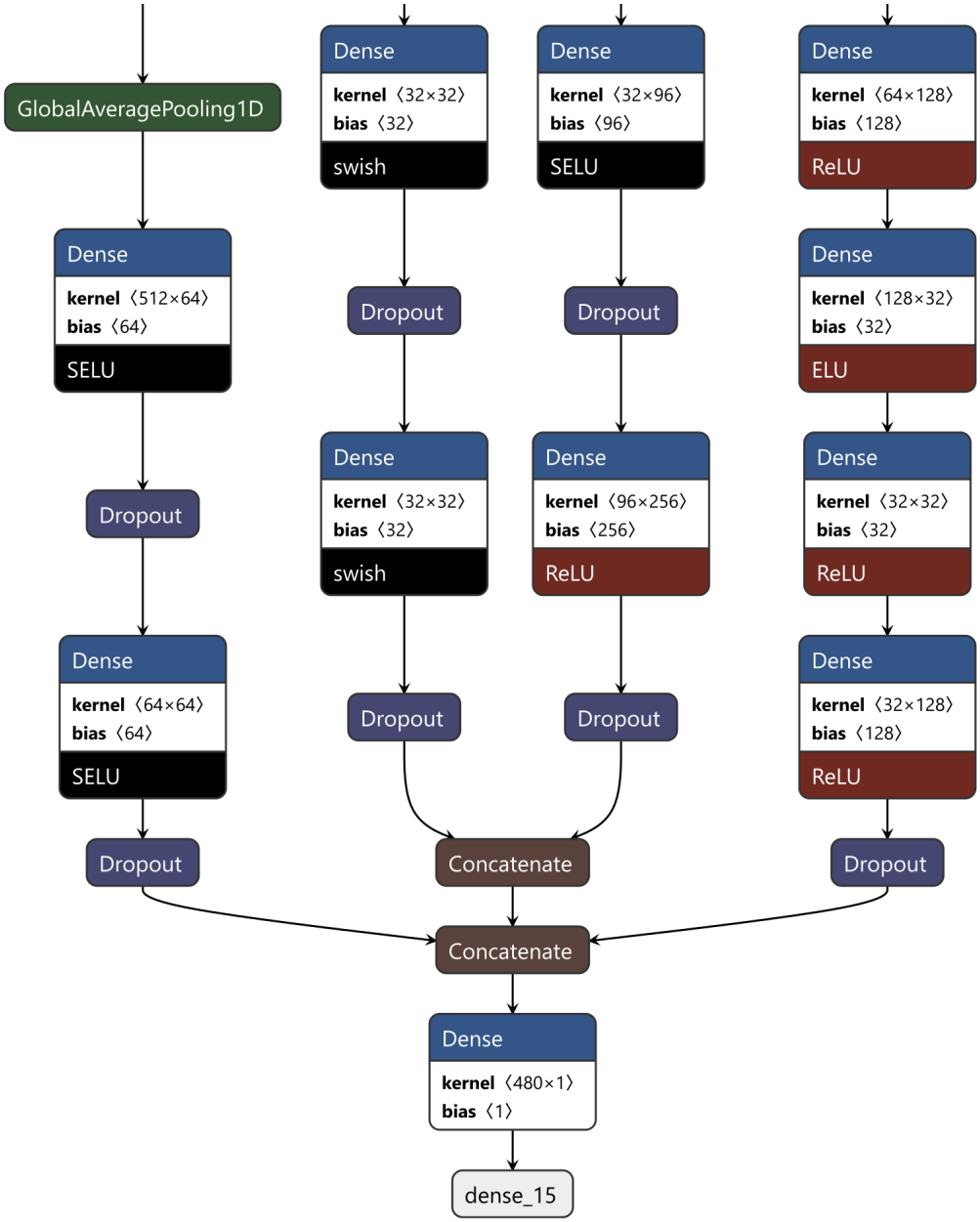


FIGURE B.6: Architecture of Hybrid Model for popularity prediction After Song Release (Part 3 of 3)

List of Figures

List of Tables