# CS137 Project: Music Popularity Prediction

Matthew Cook, Michael Loturco, and Abigail Stone

## 1 Introduction

Online music streaming services produce a weath of data about song popularity and trends in music over time. Access to this data gives us the opportunity to try to predict popularity and growth of songs. Predicting the stream count trends of a song can be advantageous for artists, producers, and streaming services. Artists may find this information useful for understanding which of their songs are gaining the most popularity, and streaming services may be able to better allocate resources to serve that song to end-users.

Existing work [2, 5] has used metadata and audio features to assign a generalized performance prediction, such as "low", "medium" and "high" popularity or "hit song" and "not hit song". The Spotify API [1] provides audio features (e.g. danciness and energy), lyrical features (e.g. valence and mood), and metadata for the track, including artist and album information. We are interested in forecasting the stream counts into future timesteps. Most of the existing literature focuses on computing a popularity score using metadata information, and we are interested in predicting stream count values over time.

Since the popularity of a track changes over time, we hypothesize that estimating "popularity" without considering the time dimension will produce poorer model performance than predicting popularity while considering time.

First, we use basic LSTM and GRU models that use time series to predict stream counts in future time steps. Then, we propose a joint architecture that considers both the encoded metadata from each track and the time series data of stream counts of those songs over time. We hypothesize that by considering stream counts over time, predictions for popularity will be more accurate.

We will evaluate our models using the standard metrics for regression problems: root mean squared error and mean absolute error. We will compare the results of our multiple-input model to the results of our own single-input LSTM and GRU models that mimic existing stream count prediction models in the literature.

## 2 Related Work

Martin-Gutierrez et. al [5] introduce the SpotGenTrack Popularity Dataset (SPD) and a deep learning architecture called HitMusicNet. SpotGenTrack combines metadata and audio data from Spotify and corresponding lyrical information from Genius and then extracts features from those sources. The extracted features are then compressed and encoded through a network MusicAENet. Finally a CNN (MusicPopNet) consumes the encoded and compressed feature vector to produce a classification of popularity. Similar previous models are noted to be binary classification, "hit" vs "not hit" whereas HitMusicNet is a ternary classifier separating "low", "medium" and "high" classes.

Li et al [4] present LSTM Rolling Prediction Algorithm (LSTM-RPA), a popularity prediction algorithm based on a Recurrent Neural Network. LSTM-RPA breaks the long-term prediction into several short-term trend predictions. This work is an important example of using an RNN architecture to predict song popularity. Wang et al [7] also present an LSTM model for forecasting popularity.

Araujo et al [2] present a popularity prediction method using Support Vector Machines; this method predicts whether a song will appear on the Spotify Top 50 Global charts. This work has similar goals to our project and will serve as an important reference for evaluating our model's accuracy.

In their 2018 paper, Choi et al [3] compare audio pre-processing methods for deep neural networks. Their primary result demonstrates that many common preprocessing techniques are redundant; this conclusion will help us inform our pre-processing decisions for our project.

# 3 Methods

## 3.1 Dataset

Our primary dataset is a modified version of the daily top 200 songs by stream count in the Philippines, from January 1st 2017 to May 20th 2021 [6]. We reshaped this dataset to be of the form $\mathbb{R}^{d \times s}$, where $d$ is the number of dates in the dataset and $s$ is the number of songs.

One issue with this dataset is that it only represents the top 200 songs for a given day; as a result, songs occasionally drop off the charts for a few days and then return, leaving gaps in our time series. To solve this, we interpolated values for gaps of size exactly one (representing one day of off-chart stream counts) and filtered out songs containing zero values greater than a strict threshold. Due to the sparsity of our data, we experimented with different thresholds for including a song resulting, in a variety of different values for $s$.

To formulate our training data, we took sub-sequences of the overall date sequence for each song, with subsequences each of size `lookback` in our training dataset and the next timestep (of length one) set as the target value.

After trimming our data set in such a manner the shape is

$$X = [\text{Batch Size, Lookback, } s]$$

$$Y = [\text{Batch Size, 1, } s]$$

where $X$ is our training data and $Y$ is our target data.

In order to test our hypothesis, one of our architectures has multiple inputs. The second data was formed by obtaining a variety of metadata for each song in the original dataset from the Spotify API. This metadata includes features such as dancibility or acousticness in the form of a real value from 0 to 1. For each of the songs, we extracted 13 metadata features and produced a tensor of shape [$s$, 13].

## 3.2 Model Architecture

The learning problem is a straightforward regression problem. Given the daily stream counts of songs over a period of time, we are looking to accurately predict the stream count of the next day. Given the time-oriented nature of the problem, our models consist of recurrent layers. We built basic models using both LSTMs and GRUs to produce baseline accuracy metrics to be used to evaluate our multiple-input models.

Figure 1: TODO: LSTM and GRU summaries

Figure 2: TODO: better-drawn version of architecture

We propose an architecture for a multi-input model that combines stream-count data with the extracted metadata. We first run the stream count information through a recurrent neural network. Then we concatenate the output from this model with the $m$ metadata features to create a feature vector of length $m + 1$. Then we feed this feature vector through a feed-forward neural network to output an improved stream count prediction.

### 3.3 Model Evaluation

We used Mean Squared Error, Root Mean Squared Error, and Mean Absolute Error as our evaluation metrics.

Additionally, we visualize the predictions of our model by plotting the predictions alongside the original stream counts.

## 4 Experiments

### 4.1 GRU and LSTM

| | LSTM | GRU |
|---|---|---|
| RMSE | | |
| MAE | | |

### 4.2 Multiple-Input Models

Our first major attempt to improve performance was to train the multiple input model, which was previously described. This did not yield satisfactory results and was not properly training.

There are two intuition as to why the model did not work. The first being that since the recurrent layers are deeper in the model then we have no control over how they train. Therefore they may not be training to predict stream count and rather some other value. The other major issue is that since the RNN is only producing one value which is being concatenated onto the metadata for each song, than the model is going to naturally give more weight to the 13 metadata variables.

## 5 Conclusion

**Something about successful vs unsuccessful models**

While we were unsuccessful in our attempts to use the metadata of the song to improve performance that may mostly be due to implementation issues. As such it could certainly benefit from further attempts. One potential modification to the multi-input model is to train the recurrent layers as a separate model for a preliminary input. Then proceed to train a model which uses that output alongside the metadata to refine the predicted value. This avoids the issue of not knowing how the recurrent layer is training in the larger model, but still may present the problem that more weight is being given to the metadata than the

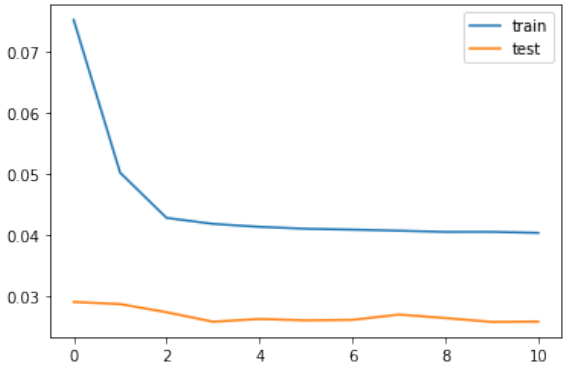


Figure 3: Training and Testing loss (MSE)

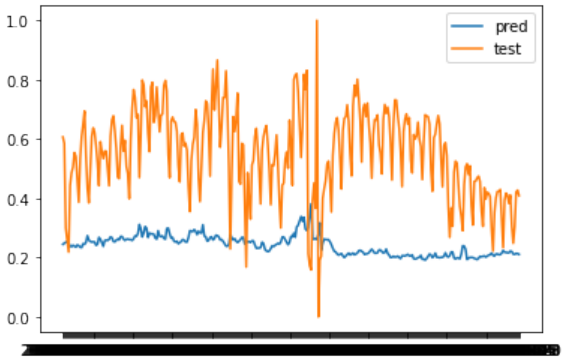


Figure 4: Predicted stream counts and validation data

predicted value. As well a larger and less sparse data set could be used a most likely would improve performance. While adding more data would improve the performance it could also lead to complications due to differing music tastes across different regions. By focusing on the Philippines data set

## 6 Share of work

## References

[1] Spotify developer API.

[2] Carlos Soares Araujo, Marco Cristo, and Rafael Giusti. Predicting music popularity on streaming platforms. In *Anais do Simpósio Brasileiro de Computação Musical (SBCM 2019)*, pages 141–148. Sociedade Brasileira de Computação - SBC.

[3] Keunwoo Choi, Gyorgy Fazekas, Mark Sandler, and Kyunghyun Cho. A comparison of audio signal preprocessing methods for deep neural networks on music tagging. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1870–1874. IEEE.

[4] Kun Li, Meng Li, Yanling Li, and Min Lin. LSTM-RPA: A simple but effective long sequence prediction algorithm for music popularity prediction.

[5] David Martin-Gutierrez, Gustavo Hernandez Penaloza, Alberto Belmonte-Hernandez, and Federico Alvarez Garcia. A multimodal end-to-end deep learning architecture for music popularity prediction. 8:39361–39374.

[6] JC Peralta. Spotify daily top 200 tracks in the philippenes.

[7] Zhenye Wang, Chengxu Ye, and Wentao Wang. Music trend forecast based on LSTM. In *2019 4th International Conference on Computational Intelligence and Applications (ICCIA)*, pages 30–35. IEEE.