CS230

Music Popularity Prediction via Techniques in Deep Supervised Learning

Brian Rossi
Department of Computer Science
Stanford University
brossi@stanford.edu

Mitch Pleus
Department of Electrical Engineering
Stanford University
apleus@stanford.edu

Abstract

In this project, we tackled the complex problem of discerning popularity of a song given its raw audio signal. This project is interesting because of its potential value to an inherently subjective industry. With a dataset of 8,000 songs and metadata including play counts for each song, our strategy was to preprocess audio files by converting them into their respective frequency spectrogram images, and use a deep convolutional neural network as a multi-class classifier to predict the audio files' popularity via their associated spectrogram images. After experimentation with different architectures and grid-search hyperparameter tuning, Our best-performing model was an 8-layer CNN with RGB spectrogram inputs that performed with 61% test accuracy, reaffirming suspicions of the ambitiousness and complexity of the problem we chose to tackle.

1 Introduction

The problem we are investigating for this project is what makes a song popular? More specifically, can we discern from a raw audio signal and its underlying characteristics whether or not a song will be popular? This project is interesting because it provides a quantitative perspective on optimizing an inherently subjective industry. Any insights into predicting the popularity of musical compositions would have high value-added applications in the music industry and advertising – such as allowing artists and labels to know which singles to push, or whether or not to keep working on their compositions.

Based on prior research into genre prediction (see "Related Work"), our approach was to convert audio signals of songs into spectrogram images. We then labeled each image with an associated popularity class (Low-Popularity, Medium-Popularity, or High-Popularity) based on the number of listens each song had. This allowed us to feed the spectrograms directly into a deep, convolutional neural network (Figure 1). Our input for each example was a scaled-down spectrogram image of size 102 x 159 x 3 (RGB images) or 102 x 159 x 1 (mono images) depending on the model architecture we were testing. The final layer of our CNN contained a softmax output in order to predict into which one of three popularity buckets each example fit. As such, our output for each example was a class label – either high, medium, or low popularity – based on the number of listens / play count of the example in question.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

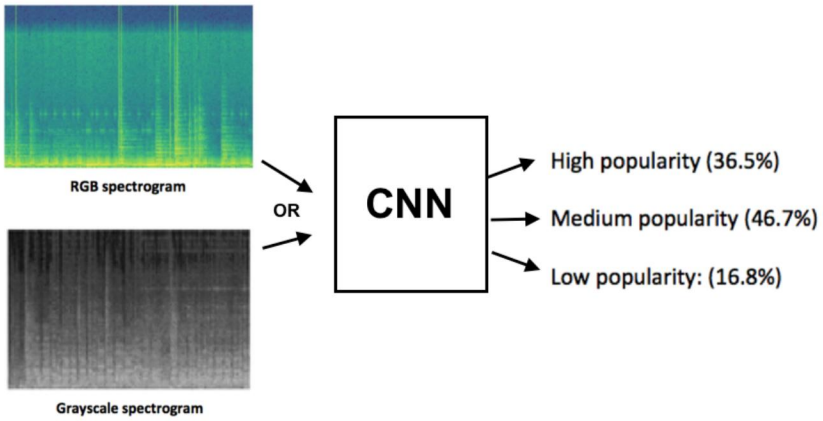

Figure 1: Inputs & Outputs. Inputs to our CNN consisted of either RGB or Grayscale spectrograms depending on our model. Our output for each example was a class label – either high, medium, or low popularity – from a softmax output.

2 Related work

A more well-researched problem than popularity prediction with more successful results is genre classification. Some papers we referenced include Meza and Nalianya's Music Genre Classification Using Deep Learning[1] and O'Beirne and Zamora's Music Genre Classification Using Mel Spectrogram Representations[2]. Both of these papers achieved their highest test accuracies (> 90%) by using the methodology and architecture of converting song audio into mel-spectrograms and feeding these spectrogram images as input into a deep CNN.

Related work on music popularity prediction includes Yang, Chou, etc.'s Problem of Audio-Based Hit Song Prediction Using Convolutional Neural Networks[3] and Pham, Kyuak, and Park's Predicting Song Popularity [4]. The first paper converted audio into mel-spectrograms in order to train different models including Logistic Regression, RNN's and different flavors of CNN's. In this paper, the deeper CNN's performed the best. The second paper used song metadata (key, loudness, mode, etc.) as input to different methods such as Logistic Regression, Linear Discriminant Analysis, Support Vector Machines, etc.

For our own project, we decided to convert raw audio into spectrograms as input to a CNN over using song metadata as input for multiple reasons. First, there is documented high performance in genre classification using this method, which is a similar application to ours, and documented higher performance using CNN's over RNN's for the problem of popularity prediction. Second, thinking ahead to applications of our project, raw audio files allowed us to create a model with more easily constructed inputs – the thinking was if our model works to a reasonable degree, it would be easier to convert and input new test songs rather than having to construct the input for each example based on a complex metadata scheme. One of the datasets we found easy access to (and ended up using) is the FMA dataset[5] which contained mp3 audio files for 100,000 songs, but coincidentally did not have a robust set of metadata features. Third, using audio to predict popularity has not been hugely successful in the past and could have groundbreaking applications to the music industry if solved. Therefore, we decided to approach our project in the same fashion as Yang, Chou, etc.'s Problem of Audio-Based Hit Song Prediction Using Convolutional Neural Networks[3] as they had a similar objective and dataset format.

3 Dataset and Features

The dataset that we are using is the FMA: A Dataset for Music Analysis[5] which is a dataset of over 100,000 sounds (mp3 format) and their associated metadata. For our project (due to space and time constraints), we are using only a subset of their data, namely, the fma-small dataset which includes

8,000 tracks of 30 seconds a piece. From this data and the associated metadata, we are able to match each track to a listens count, which we used as our basis for labeling popularity.

Because our CNN is taking images (spectrograms) as input, there is a fair amount of data preprocessing that must be done. After downloading all of the data, we had to convert all mp3 files to .wav files, and then from .wav files to .png spectrograms (Figure 2). To compress the data, we cut the spectrograms down to a random 10s sample from each 30s clip. We felt this was a reasonable unit of time for a human to discern whether or not they liked the song i.e. predict popularity. We also downsized the image from size [1005 x 306 x 3] to [102 x 159 x 3], and then converted our spectrograms to grayscale to cut down computation and achieve reasonable training times for our model (Note that we ended up training some models taking RGB input and some taking mono input).

We then decided to label our examples into 3 classes: low-popularity (16.8% of total dataset), med-popularity (46.7% of total dataset), and high-popularity (36.5% of total dataset) based on each example's associated #listens count in its metadata. We decided to create these 3 buckets because we believed it would be a more intuitive metric to evaluate how our model was doing vs. trying to predict an actual listens score in the same format as our original dataset's metadata.

Lastly, we split our dataset into an 80%, 10%, 10% split corresponding to train, validation, and test datasets with the same distribution of popularity classes in each set.

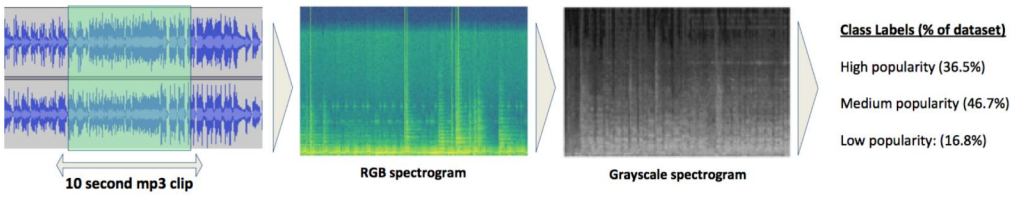

Figure 2: Data preprocessing. For each example, we 1) randomly sampled 10 seconds from the given 30 second audio clip, 2) converted the audio clip to its RGB spectrogram, 3) downsized the image to 102 x 159 x 3, 4) converted to grayscale / mono, 5) bucketed its #listens metadata to find its associated class label out of high, medium, and low popularity.

4 Methods

The four main models that we trained and tuned were a 4-Layer CNN with mono spectrogram input, a 6-Layer CNN with mono spectrogram input, a 6-Layer CNN with RGB spectrogram input, and an 8-Layer CNN with RGB spectrogram input.

For each of our different models, we used a CNN with Adam Optimization and mini-batches. Each convolutional layer had ReLU activation functions and used batch normalization as well as a maxpool layer. Our first FC layer used dropout so as to prevent overfitting and our output layer was a softmax layer to predict one of three popularity classes.

For our cost function, we used the mean of the softmax cross entropy between the logits we computed from our CNN and the true data labels.

$$-\sum_{c=1}^{M} y_{o,c} * log(p_{o,c})$$
$$M = 3\, classes$$
$$y = true\ label$$
$$p = probability\ example\ o\ is\ of\ class\ c$$

5 Experiments/Results/Discussion

When training each of our models, we decided to tune learning rate, mini-batch size, and dropout rate. We conducted a grid search strategy for our hyperparameter tuning. We conducted hyperparameter tuning for each of our 4 main models.

We evaluated our models based on the train and test accuracies of the outputs. If the model outputted the correct class (low, medium, or high popularity) for an example, then we would count that as a correct prediction and an incorrect prediction otherwise. We chose to use this as our performance metric because we believed it aligns potential evaluation of human performance on this task.

After training our earlier models (Figure 3), we noticed that we were underfitting. Our approach was to see if deeper networks as well as incorporating RGB spectrograms as input instead of mono would lead to higher training accuracies.

Accuracies			
Architecture	**Training**	**Validation**	**Test**
2-Conv Layer CNN (mono)	44%	41%	42%
4-Conv Layer CNN (mono)	46%	51%	51%
4-Conv Layer CNN (RGB)	53%	55%	53%
6-Conv Layer CNN (RGB)	59%	63%	61%

Figure 3: Our models and their associated train, validation, and test accuracies.

This led to our best-performing model which was an 8-Layer CNN with RGB inputs (Figure 4). We were able to achieve 61% test accuracy on this model and the associated tuned hyperparameters are shown in figure 5.

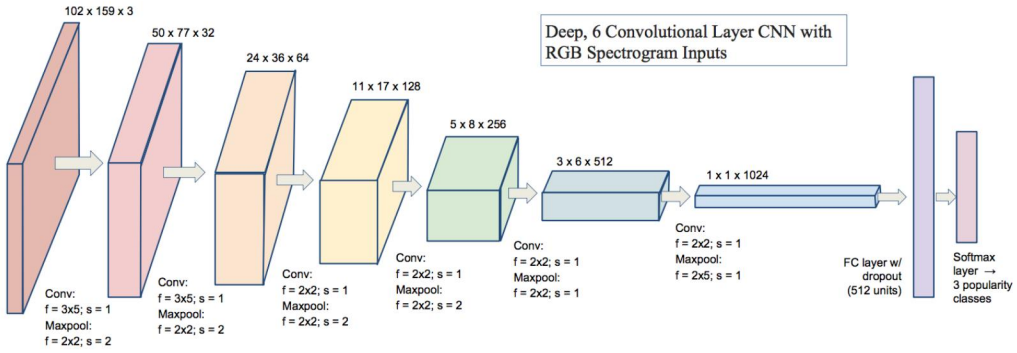

Figure 4: Best performing 8-layer CNN dimensions and parameters, with 61% test accuracy. 6 convolutional layers, a fully connected layer with 512 units, and a softmax layer with 3 popularity classes.

Hyperparameters for Best Model		
Learning Rate	**Batch Size**	**Dropout Rate**
5 e-4	16	0.4

Figure 5: Hyperparameter values for best performing model.

As seen by the results in figure 3, the more information we incorporated into the input and the deeper we made the model, the better the performance. This is likely due to the complexity and subjectivity of the task at hand. More complex inputs gave our network more information per example with which to learn, and the deeper our network, the more expressive features it could extract from the data.

When we saw the model was underfitting, we capped our number epochs at 100 and focused more on changing our model architecture to deeper networks with more expressive inputs. This is because we saw the loss function and accuracy flatten out and deduced that our network wasn't better learning features from increasing the training time.

6 Conclusion/Future Work

From this project, we see that it is very difficult and ambitious task to predict the popularity of a song, especially solely from the raw audio signal. In training our CNN, we battled against underfitting, which is why in our later models, we moved to deeper networks and transitioned from mono input to RGB input. This led us to our best-performing model which was an 8-layer CNN giving us a 61% test accuracy.

Underfitting leads us to believe future models might benefit from more expressive information per example in our dataset. In future work, it would be interesting to see if incorporating more discrete metadata (e.g. artist popularity, genre, tempo, key, etc.) into our model would increase accuracy. Though it was tough given the associated metadata of our dataset, using a more vanilla neural network in parallel with our CNN analysis might provide more expressive input data, and consequently more accurate results. Taking a larger sample of audio per example (vs. the 10s we used for this project) could also lead to a higher prediction accuracy.

Spectrograms can theoretically glean harmonic and rhythmic information, but more in depth analysis of different musical touchstones and characteristics could supplement future popularity predictors (e.g. NLP analysis of lyrics).

7 Contributions

Both members have researched existing implementations and literature in order to plan the project. Mitch was responsible for converting from original mp3's to .wavs to spectrograms. Brian was responsible for data collection as well as data formatting once spectrograms were computed and hyperparameter tuning and analysis. Mitch performed majority of implementation of baseline CNN models, developed while referencing the Tensorflow Tutorial from coursera as well as CNN starter code provided by TA's. Both members worked in conjunction to discuss ways of improving earlier models and implementing deeper networks with RGB inputs.

References

[1] Meza, M. & Nlianya, J. (2018) Music Genre Classification Using Deep Learning.

[2] O'Bierne, A. & Zamora, A. (2018) Music Genre Classification Using Mel Spectrogram Representations.

[3] Yang, L. & Chou, S. & Liu, J. & Yang, Y. & Chen, Y. (2017) Revisiting the Problem of Audio-Based Hit Song Prediction Using Convolutional Neural Networks.

[4] Pham, J. & Kyuak, E. & Park, E. (2015) Predicting Song Popularity.

[5] Defferrard, M. & Benzi, K. & Vandergheynst, P. & Bresson, X. (2017) FMA: A Dataset for Music Analysis.