Evaluating Techniques for Predicting the Trending Year of a Song

Horatiu Lazu, Ryan Kim, Peter Huang
{hlazu, ryan.kim, zh3huang}@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Abstract

The purpose of this paper is to explore the possibility of applying artificial intelligence (AI) and machine learning algorithms such as Convolutional Neural Networks (CNN) and classical machine learning models to predict the trending year of a song. This will allow music streaming applications to have the ability to suggest a wider variety of music than it previously could. Music listeners would be able to enjoy curated playlists of songs from a specific range of trending years, such as the 90's. This project compares the performance across different models such as CNNs, linear regression, support vector regression, decision trees, and random forests. Each model was evaluated based on their prediction accuracy for the trending year of a song. The results show that the CNN model performed the best compared to the other classical machine learning models, with an average error of 7.2 years when predicting songs between 1960 and 2020.

Introduction

Music classification is a popular topic among applications of AI and machine learning in the music industry. The idea is to automatically classify music based on musical genre, tempo, rhythm, beat, and/or lyrics. This classification is then used by products such as Spotify and Apple Music to recommend similar songs to users, given their preference of music. Before methods based on AI and machine learning, playlist curators would manually curate their own music playlists to distribute to others.

Though music classification by musical genre has been heavily explored by various research papers and implementations, not many have tried to apply the same AI and machine learning techniques to predict what time period a song sounds like it is from. As humans, we can tell the between songs that were popular in the 60s compared to songs that popular now, but no research has applied AI and machine learning techniques to allow a machine to differentiate between music of certain time periods. Using new models, music recommendation systems can suggest a larger category of suggestions than they previously could. For example, someone may want to listen to music that sounds like it is from the 60's even if the song was released much later. With music classification based on trending year, people would be able to find songs and bands that resemble the sound signatures of a certain time period, despite when the song was actually released. Looking at the ID3 metadata tags in the audio files will mention what year the song was released, but this may not be correlated to what year the song was popular. Also, the metadata tags stating the release year of a song would not be able to determine if a modern song released in 2020 sounds like it is from the 1960s. This would allow music providers such as Spotify to suggest a larger pool of songs that resemble the sound signature of a time period. Someone who is interested in listening to rock music that sounds like it is from the 60s would be able to find modern bands that release new songs that resemble the sound of 60s rock.

From a commercial standpoint, this is an important problem to tackle because it would allow for better recommendation of music. If people are recommended music that more closely matches their tastes, they would discover songs that they are more likely to listen to. Companies such as Spotify currently use Convolutional Neural Nets (CNNs) to hone their recommendation systems (Tyagi 2020). These companies offer AI driven recommendation systems as a feature to attract customers.

From a cultural standpoint, this is an important problem to tackle because it would allow us to analyze trends in music. For example, if we look at western popular music in the 1960s, The Beatles and their style of music was extremely popular. Not only that, but they had profound cultural influences on society as a whole. Being able to analyze these trends would help us gain insights and reasoning into what kind of music was popular during a specific time period.

The purpose of this project is to evaluate and compare the performance between Convoluted Neural Networks (CNNs) and classical machine learning method such as decision trees, random forests, linear regression, and support vector regression when they are made to predict the trending year of songs. Given a song, the models should be able to determine what year the song likely would have been popular in. In any given year, there are many musical genres with greatly differing tempos, beats, rhythms, vocals, and lyrics. A successful model for classifying music based on trending year should be able to accurately distinguish between music such as music from the 60's and music from the 80's. The current models for genre classification would only be able to differentiate between different types of genres such as rock or country, but not the specific time period that the music is from. The models should work for historic songs, such as songs that were actually popular during a certain time period, and it should also work for new songs which sound like they are from a certain time period. This is a fundamentally more difficult problem compared to genre classification because songs that are popular in any given year can be from a variety of genres.

The methodology would involve building each of the models (CNN, decision tree, random forest, linear regression, support vector regression), and testing them against the dataset set. The dataset would be acquired by looking at the billboard top 100 songs per year (BillboardTop100 2020), since that would provide the songs that were trending during a time period. The songs from the billboard will then be downloaded through proprietary means. The songs cannot be shared due to copyright issues. However, all of the songs are available on YouTube.

Existing Github repositories for music genre classification will be utilized. This includes the gtzan-keras repository (Guimaraes 2020) and the music-genre-classification repository (Cetin 2019). However, these repositories need to be modified. In particular, the logistic regression will be changed to linear regression, since time is a continuous value whereas genre is a discrete value. The support vector machine will be changed to a support vector regression, for the same reason just stated. The decision tree and random forest implementations will also need to be modified to support continuous rather than discrete values. The CNN will use a slightly different structure. For example, the last layer not having as many out channels as genres but instead having only one to indicate the predicted trending year.

The performance measure for the comparison of CNNs and classical machine learning methods such as decision trees, random forests, linear regression, and support vector regression will be the Mean Absolute Error (MAE) formula, given by Figure 1. Each of the songs in the testing set will be run against each of the models. Each of their errors, given by $|actual\_trending\_year - predicted\_trending\_year|$, will be summed up and divided by the total number of songs in the testing set. This should give a reflection of how accurate the models are when predicting the trending year of a song. A lower number would indicate a better model, and a higher value would indicate a worse model.

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Figure 1: Mean Absolute Error (MAE)

The key findings show that CNNs perform much better when predicting the trending year of a song compared to classical models such as linear regression, support vector regression, decision trees, and random forests. The CNN model was able to predict the trending year of a song within 5 years of the actual year, 53% of the time. On the other hand, the classical models were only able to predict the trending year of a song within 5 years of the actual year, 26% to 36% of the time. Loosening the time range, the CNN model was able to predict the correct trending year within 10 years of the actual year 83% of the time, whereas the classical models were able to predict the correct trending year range 47% to 65% of the time. These are very promising results for the CNN model.

Although many previous works have tried to classify songs by genre using various machine learning models such as CNNs, linear regression, support vector regression, decision trees, and random forests, none have tried to apply these models for predicting the trending year of a song. This is a much harder problem because the input signals are much noisier because there are many genres for each trending year, and songs of similar genres across different trending years may not have a clear distinction. For example, music genres typically have a general tempo which is consistent throughout songs in a given genre, however this is not true for songs of any given year. A slow and romantic song could be a top 10 trending song in one year, and a fast-beat metal song could also be a top 10 trending song in the same year. This makes the signal fundamentally noisier compared to classifying songs by genre.

Compared to the problem of music genre classification, this problem is a regression problem rather than a classification problem. Previous works done for music classification had to be reworked to apply to regression rather than classification. For example, logistic regression had to be changed to linear regression and support vector machines had to be changed to support vector regression. In addition, while there is a popular dataset called the GTZAN dataset for music genre classification, there is no such widely available dataset for trending year prediction. This dataset had to be collected through proprietary means.

The results show that machine learning and AI can be used for predicting the trending year of a song. In particular, the CNN model was very accurate in predicting the trending year of a song, and would likely predict the release year with almost the same degree of accuracy as a human.

Some key contributions include:

- First application of machine learning and AI techniques for music trending year prediction
- First comparison of performance between CNNs and classical approaches for music trending year prediction
- First exploration of regression techniques for music trending year in musical datasets

Related Work

The general area of MIR (music information retrieval) focuses on retrieving information from music, such as artist and album. Over the years, there have been many papers released in the field of music classification with common themes focusing on genre, style and mood. The landmark paper of Dannenberg, Thom and Watson is particularly well-known for establishing the baseline in musical style classifiers. Over the years research expanded into more complex feature engineering leveraging audio signaling, statistical and information theory techniques, in addition to use of deep neural networks. There is no significant research in the area of classifying the release year or decade of a song.

At first, the development of musical style classifiers was explored using naive Bayesian, linear and neural network approaches (Dannenberg, Thom, and Watson 1997). Styles were defined based on performance intentions (such as high, low, quote, or blues). Neural networks posed to be the most powerful because it did not assume any underlying probability distributions and can incorporate non-linear terms. However, they performed worse than the other two as the number of classification classes increased.

Classifiers	Bayesian	Linear	Neural Network
4	98.1	99.4	98.5
8	90.0	84.3	77.0

Figure 2: Percentage of Correct classifications (Dannenberg, Thom, and Watson 1997)

Following the original landmark paper, subsequent research focused more specifically on genre classification. The underlying approach of most use feature extraction on timbre texture, rhythmic and/or pitch content (in some combination).

Tzanetakis and Cook's paper (Tzanetakis, Essl, and Cook 2002) proposes using a variety of feature extraction types, including musical surface features (characteristics related to the texture, instrumentation and timbre of the sound), rhythm and pitch features. They combine the feature sets, along with applying Short-time Fourier Transform (STFT), Mel-frequency Cepstral Coefficients (MFCCs), Wavelet Transform (WT) and additional parameters to create the feature vectors. Gaussian classifiers were trained where each of the classes was represented using a single multidimensional Gaussian distribution with the parameters determined from the training set. The results were positive, with the musical surface being shown to provide a powerful signal for training the models. Figure 3 outlines combinations of music genre as columns and the percentage guessed as the genre in the associated row, for example percentage rock guessed as hiphop is 18%. Figure 4 outlines the impact different feature types impact correct genre classification.

	Classic	Country	Disco	Hiphop	Jazz	Rock
Classic	86	2	0	4	18	1
Country	1	57	5	1	12	13
Disco	0	6	55	4	0	5
Hiphop	0	15	28	90	4	18
Jazz	7	1	0	0	0.37	12
Rock	6	19	11	0	27	48

Figure 3: Genre classification confusion matrix (Tzanetakis, Essl, and Cook 2002)

The growing amount of interest in music genre classification led to a more rigorous comparison between the different content-based music genre classification techniques

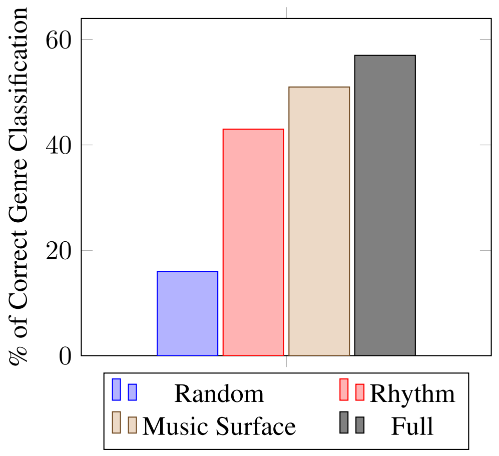

Figure 4: Relative feature set importance (Tzanetakis, Essl, and Cook 2002)

(Li, Ogihara, and Li 2003). Different content-based features were evaluated, such as rhythmic and pitch, in addition to a novel set known as DWCHs which is based on wavelet histograms and captures both local and information of audio signals at the same time. According to the results, it was found that Melfrequency Cepstral Coefficients (MFCCs) and timbral texture features excluding MFCCs were most effective and rhythm/pitch being less effective. It was also observed that the DWCHs feature vector was the best performing, along with the SVM being the best model for the results.

Instead of employing musical theory approaches to feature extraction, Antonio Jose Homsi Goulart et al explored applying information theory (entropies and fractal dimensions) instead (Goulart, Guido, and Maciel 2012). First, entropies were created by dividing audio files into frames (each having 1024 samples and with 50% overlap between frames), and computed entropy based on a specific formula. Feature vectors consist of different statistical properties of the frame's entropies (average entropy, standard deviation, maximum, minimum, range) along with fractal dimensions of each frame (created by a technique known as box counting method). The classification approach creates a combined SVM, with each SVM determining if the audio track corresponds to its respective genre. The results from benchmarking indicated that fractal dimensions did not add accuracy to the test results, and that by having a very large amount of training data the models produced very accurate models with upwards of 90% accuracy.

In recent years, deep neural networks have been at the forefront of musical genre classification. Convolutional Deep Belief Networks (CDBN) were shown to improve music genre classification performance by leveraging audio spectrogram and MFCC features (Zhang et al. 2016). It was later shown by Weibin Zhang et al that CNNs for music genre classification combined with shortcut connections for skipping layers and using an approach for "average/maximum pooling" to provide statistical information to higher-level neural networks allowed for increased accuracy of the genre classifiers, leading to over 80% accuracy.

Overall, the area of genre classification was deemed a difficult problem. Cory McKay and Ichiro Fujinaga argue that genre classification is hindered by the fact that genres aren't strictly defined for many songs, and in fact songs may belong to more than one genre (McKay and Fujinaga 2006). In addition, an often-cited study found that a group of undergraduate students agreed with record companies only 72% of the time. It was also found that powerful machine learning algorithms commonly used such as AdaBoost and SVMs do not scale well. Very similar arguments can be presented for classifying songs based on release year/decade, and will likely be a challenge for producing useful results.

Methodology

The algorithms used for predicting the trending year of a song will be Convolutional Neural Networks (CNNs), and classical machine learning algorithms such as decision trees, random forests, linear regression, and support vector regression. As mentioned in the related works, neural networks posed to be the most powerful for classifying music because it does not assume any underlying probability distributions and can incorporate non-linear terms. The CNN algorithm will be compared to the classical machine learning approaches for predicting the trending year of a song. The reason behind choosing these specific algorithms is that CNNs and the classical methods are prominently used in related works and open-source projects.

Convolutional Neural Networks (CNN) are a deep-learning algorithm used for both classification and regression problems, first introduced by Yann LeCun in the 1980s (LeCun et al. 1999). CNNs are the result of combining neural networks and convolutions. Based on neural networks, an artificial neuron is defined as a node consisting of a series of inputs and a bias each associated with a weight, along with an activator function that takes those weighted inputs and produces a result. The network comprises of numerous layers, including one or more steps of convolution. The convolution and pooling steps will perform feature extractions and reduce the dimensionality of the data, while the later steps will apply classification/regression algorithms to produce the final result.

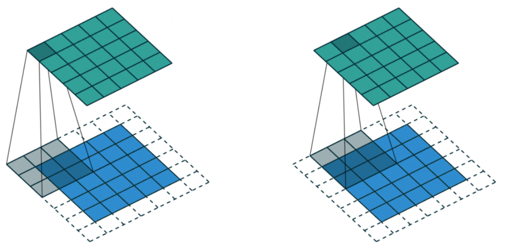
Figure 5: Convolution Example (Shafkat 2018)

Convolution steps involve convolving chosen filters on the input (by sliding the filter across the input matrix and applying a dot product, see Figure 5), with each filter producing a new resulting output (known as tensor). The role of the convolutions are to extract features from the inputs, but preserve the spatial relationship between the elements in the matrix. The tensors formed by the kernels will then be the input for the next set of convolutions, and this process can be repeated several times with different filters. After each convolution, a non-linear operation (for example ReLu, rectified linear unit operation which is the function of max(0, value)) may be applied for all values in the matrix. Applying such an operation will make the model perform non-linear operations, because otherwise all linear combinations of a linear function would still be linear, opening the door for more complex representations. Note however that the non-linear operation may also be present in the later ”fully-connected layer” step.

The next layers will consist of pooling, which aims to reduce the size of the data. This is achieved by defining a spatial neighborhood window (such as a 2x2 matrix), partitioning the original matrix into a series of windows and for each window performing an operation (such as taking the maximum, average or summation of all its values). This results in a reduction in the input representation while preserving important qualities of the data, and makes the network less susceptible to small distortions such as value translations. The last pooling layer is an input for the fully-connected layer, which consists of a traditional neural network where each input and bias has a weight and the output from it is the final result. Weights for the fully-connected layer can be determined using forward and backpropagation, indifferent of a traditional neural network. The final layer of the fully-connected network is a classifier or regression function that will produce the final result. SoftMax is a popular choice for an activation function, however in the context of regression it isn't very useful because SoftMax assigns decimal probabilities to different classes with a sum up to one (Google 2020). Instead, using a ReLU for the activation function would be a better choice for this problem, because all predictions should be positive.

In the field of artificial intelligence, decision trees are one of the simplest and most successful forms of reasoning. Decision trees represent functions that take on a list of continuous or discrete attribute values as the input, and subsequently returns a single continuous or discrete decision as the output. In the process, the decision tree's function will perform a sequence of tests against the input. Visually, each node in the decision tree represents the test result of an input attribute. Additionally, the branches connecting the nodes each represents a possible value of the input attribute. Finally, the leaf nodes of the decision tree are the possible values that could be returned by the decision tree function (Russell and Norvig 2016). Nonetheless, decision trees are very intuitive for humans as they can be found in many troubleshooting or tutorial guides.

Part of the challenge when applying the use of decision trees to an artificial intelligence problem is that the decision tree can be huge and results could be inconclusive without sufficient data. For each test that we apply on the input attribute data, we want to maximize the information gain by reducing the associated uncertainty in the variable or the entropy. With continuous input data, using a well placed split point achieving the highest information gain would be much more suited over generating infinitely many branches for each input value. Well ordered tests with sufficient test data yield more efficient decision trees, as likewise, techniques such as decision tree pruning and early stopping can be used. This eliminates nodes which are not clearly relevant and stops generating nodes when it is clear that there is no good input attribute to split on (Russell and Norvig 2016).

Building upon decision trees, a random forest is made up of a set of decision trees. Each decision tree in the set is trained with a random subset of the training data. The number of trees, the number of test data and the number of tests for each tree can be specified. Next, for each tree in the set, the specified number of tests against the input attributes will be selected, and the specified number of test data will be sampled with replacement. This is also known as the bootstrap aggregation (bagging) technique. Each tree will be trained in parallel and the random sampling adds the extra element of randomness that prevents overfitting. Once the decision or the predicted outcomes from each tree in the random forest is available, the results will be aggregated through model votes or taking the average, and finally deduced to formulate a single decision (Chakure 2019).

Linear regression is a machine learning algorithm where the target output is continuous and has a constant slope. The training data is used to draw a slope to predict future output using that slope. The slope is drawn such that the error rate is minimized, where the error rate is defined by measures such as mean absolute error, mean square error, mean absolute percentage error, or mean percentage error. For the case of music classification, multivariate linear regression will be used, because there are many features that are used as inputs. Multivariate regression is similar to simple regression other than that it uses a more complex multi-variable linear equation rather than a simple slope-intercept form equation. Previous works regarding genre classification of music use logistic regression rather than linear regression since genres are discrete. However, since time is a continuous value, linear regression is used rather than logistic regression.

Support Vector Regression (SVR) is similar to Support Vector Machines (SVMs), which are commonly used for classification problems in machine learning. However, the difference is that SVRs output continuous values rather than discrete values. This is what we desire given that we want to output years instead of genres. SVR follows the same basic principles as SVM. In SVM, a line is drawn to separate the data into different groupings, in potentially multidimensional space. The difference with SVR is that this line, also known as the hyperplane, is used to represent data within the margin of error, rather than dividing the data into separate groups. This is similar to linear regression, except it uses two boundary lines to fit the error within a certain threshold, rather than trying to minimize the error for all data points (as shown in Figure 6). In short, rather than trying to minimize the error rate like with linear regression, the data is fitted such that most of the points lie within the boundary line. Some key hyperparameters of SVM and SVR include the kernel, which determines the dimension of the line, the hyperplane, which is the line, and the decision boundary, where values that lie on the line may be on one side or the other.

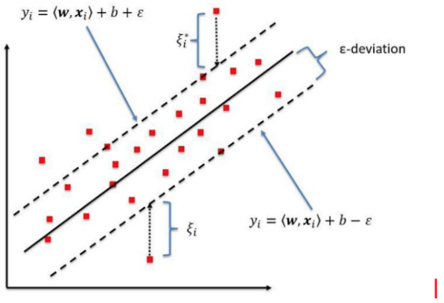
$y_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + b + \varepsilon$
ξ_i^*
ε-deviation
ξ_i
$y_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + b - \varepsilon$

Figure 6: SVG Example (Bhattacharyya 2018)

Audio does not have a convenient matrix representation such as an image for providing as input to a CNN. For applications in audio-processing, melspectrograms are commonly used (ICLR 2018). First, the Fourier Transform algorithm can be used to convert the audio signals from time domain to frequency domain, producing a result known as spectrum. However, since the sounds are typically non-periodic in music, a short-time Fourier transform is used to compute spectrums on multiple windowed segments of the same signal. The Mel scale is used to adjust for human hearing capabilities, as it was found that humans can hear lower (500-1000hz) frequencies with better accuracy than higher (1000hz+); hence the mel scale was defined to transform the spectrogram into a melspectrogram where the frequencies are converted to account for such a fact. Alongside melspectrogram, STFTs are another option, but it was found that they typically required more training data to perform as well as melspectrograms and hence melspectrograms are a good choice for this project because we are limited in number of songs to train with (Choi et al. 2018).

Unlike the CNN where we take the melspectrogram of the audio, classical machine learning models such as decision trees, random forests, linear regression, and support vector regression cannot make sense of the data contained in a melspectrogram. Thus, we need to extract various features from the audio. These features include spectral centroid, spectral rolloff, zero crossing rate, root mean squared, onset strength, spectral contract, spectral bandwidth, and spectral flatness, and tempo. The features are explained below in Figure 8. These are features that are commonly chosen for genre classification, since they provide good characteristics for distinguishing between different sounds, as mentioned in the paper by Tsinghua University ”Therefore, many previous researches turned to spectral characteristics, which are found useful for discriminating different music types and easy to be extracted” (Jiang et al. 2002). A limitation of this, however, is that the features that provide a good signal for genre classification may not provide as good a signal for predicting the year a song was trending. For example, tempo is likely to show a greater correlation with the genre of a song rather than with the release year of the song, simply because the tempo is generally consistent among songs from the same genre. Further experimentation is required to determine which features provide a good signal for predicting the year a song was trending. One feature that was left out was the lyrics of a song. This is because the lyrics of a song may not be indicative of when the song was written, especially when covers of the same song are made popular much later after its initial release.

As mentioned in the paper by Tsinghua University , ”There are many music characteristics could be used to discriminate different music type, such as the musical structure, tempo, rhythm, melody, chord, and so on. However, it is extremely difficult to extract them precisely by signal processing methods for most digital music.”(Jiang et al. 2002). The same reason applies to the reason why most of the music characteristics were not used as features for training the models for predicting the year a song was popular. Spectral characteristics shown were used instead.

Feature Name	Description
Spectral centroid	Location of the center of mass of a spectrum
Spectral rolloff	Frequency below which a specified percentage of the total spectral energy lies
Zero crossing rate	The rate at which a signal changes from negative to positive or vice versa
Root mean squared	The square root of the mean squared of the signal amplitude
Spectral flux (onset strength)	Measure of how quickly the power spectrum of a signal changes
Spectral contract	Spectral characteristic created based on the spectral peak, spectral valley and their difference in each sub-band, see (Jiang et al. 2002)
Spectral bandwidth	The width of the band of light at one-half the peak maximum
Spectral flatness	Measure of how tone-like a sound is
Tempo	Tempo of the song

Figure 7: Features extracted for the classical models

For the implementation of CNNs and the classical models, pre-existing GitHub repositories will be used. The gtzan_keras GitHub repository (Guimaraes 2020) and the music-genre-classification (Cetin 2019) provide good implementations for genre classification. Several modifications are required to work with time rather than genres. For example, Logistic Regression needs to be changed to Linear Regression, Support Vector Machines need to be changed to Support Vector Regression, and the CNNs need to support continuous output values rather than discrete output values.

The dataset was created by looking up the billboard top 100 songs per year from 1960 to 2020 using (Billboard-Top100 2020). These songs are representative of the sound signature of a certain time period in the US because these were the most popular songs during that time period in the US. From the billboard top 100, the songs were acquired through proprietary means, though they cannot be shared due to copyright issues. However, each of the songs in the billboard 100 list can be accessed through YouTube.

The reason this dataset was created was because there is no other openly available dataset that provides a list of top 100 songs from 1960 to 2020, due to copyright issues. For genre classification, there is a popular dataset known as the GTZAN dataset (Sturm 2013), which contains a collection of songs categorized by genre. Many research papers related to music genre classification use this dataset, but it would not be helpful for building models to predict the time period that a song was popular in since the dataset is only categorized by genre. Also, there are many flaws with the dataset, such as duplicated songs, as mentioned on the GTZAN dataset page (Sturm 2013).

Since the top 100 songs were collected from 1960 to 2020, a total of 6100 songs were used for the dataset. To preprocess the data for the CNN, each of the songs were split into 20 different slices which are each individually fed into the CNN. These smaller slices are easier to classify due to providing a cleaner signal for training the model. Also, each of the features extracted for the classical machine learning models are normalized to values between 0 and 1 since the models are sensitive to feature scaling. For the CNN, the monaural audio signal is extracted and log scaled.

Taking the top 100 songs from the billboard every year is reasonable for our research question because songs that make the billboard are popular and trending in that year, which is why they are on the billboard. This more accurately represents the sound signature from a certain time period compared to a random selection of music for a given year. This dataset is balanced because exactly 100 songs are taken from every year to train the models. However, the dataset is skewed towards pop music. There are vastly many types of music that do not make the billboard, and all of these would be neglected. Another limitation is that the dataset represents only the popular songs in the US, and not other parts of the world.

Results

For the evaluation of the algorithms, the CNN model will be compared to the classical models (decision tree, random forest, linear regression, support vector regression). The performance measures of our models will be determined using Mean Absolute Error (MAE), also known as the L1 loss function or Lasso. The function will take the absolute difference between the estimated trending year and the actual trending year for a respective song. Then, all of the absolute differences are added up and divided by the total number of songs predicted. This is given by the formula below. This final number is used as the benchmark across the different models. Since this is a supervised model where the exact year a song was trending is known from the billboard, it is an objective measure of performance that determines how close the predictions are to the actual value. Several alternative measurement techniques were considered, including L2 (Mean Squared Error), Huber Loss and Log-Cosh Loss. The formulas for these alternatives are given in Figure 8. The justification behind choosing MAE is that it is simpler to interpret and provides a straightforward numerical meaning of ”the average number of years the prediction is off from the target”.

Mean squared error (MSE)	$\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2$
Mean absolute error (MAE)	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i-\hat{y}_i\rvert$

Figure 8: MSE and MAE equations

We will perform a 60/20/20 train/validation/test split across the dataset, which is recommended according to Andrew Ng (Ng 2017). The members of the sets are determined randomly with a consistent seed to promote reproducibility. We do not plan on using a k-fold validation split because the original model did not, and the split is sufficient for our purposes to avoid overfitting when benchmarking our models. The 60/20/20 split will be selected per year, such that there are exactly 60 songs for training, 20 songs for validation, and 20 songs for testing per year of the dataset. This is a better approach compared to selecting the 60/20/20 split based on the total number of songs (6100) because the random selection may result in an imbalanced selection of songs used for training/validation/testing in a given year. For example, certain years may end up with more than 60 songs in the training set while other years end up with less. Our choice of selecting the split per year will ensure that the dataset will always contain exactly 60 songs in the training set, 20 songs in the validation set, and 20 songs in the testing set per year.

Once the training is done, the songs in the testing set will be split into 20 equal sized chunks. For each of the chunks, they will be fed into the CNN or classical models. This outputs the predicted year for when the song chunk was trending. As a measure of central tendency, median is chosen across all of the different chunks to produce the final result (where the average of the two middle elements is chosen in the case of a tie). Median is chosen over mean because it is less susceptible to outliers. Then, the results from the running the testing sets against the models will be compared. This experimental setup should allow for an objective comparison on the performance of each of the models for predicting the trending year of a song. The trending year of the song is already known from the billboard 100 list, so a lower MAE value should indicate a better prediction.

The hyper-parameters for the CNN that will be used are shown in Figure 20. The hyper-parameters for the classical models will be selected using the GridSearchCV function in the scikit-learn library. GridSearchCV, using the parameters that have been specified in the tables, will automatically tune the hyper-parameters for the given problem. GridSearchCV performs an exhaustive search over specified parameter values for an estimator (model). It loops through predefined parameters for the given model and provides the best fit model based on the training set. From the output of GridSearchCV, the best hyperparameters can be selected. The parameters for GridSearchCV are shown in Figures 21-24.

CNNs are expected to work better because they do not assume any underlying probability distributions and can incorporate non-linear terms . Also, classical machine learning methods such as decision trees, random forests, linear regression, and support vector regression, with human chosen features (such as what we have done), are expected to miss out on key characteristics of songs that are critical for differentiation, as mentioned in a Stanford University paper (Li, Xue, and Zhang 2018). The melspectrogram approach taken by the CNN experimental model should allow the neural network to capture key differentiating characteristics about the trending songs of a time period better than the human selected features that are used with the classical models.

CNNs have been shown to work better for genre classification compared to the classical approaches, and this is expected to extend to trending year prediction as well. However, both the CNN and classical models are expected to perform much worse at trending year prediction compared to genre classification. This is due to the increased noise in the training data. The noise would be the result of the wider variety of musical sounds of trending songs, compared to a more well-defined classification for genres. Songs from the same genre are expected to be a lot more similar to another compared to songs from the same time period. In a certain time period, there may be numerous genres and a wide variety of musical styles that are popular. Thus, the trending year prediction is expected to perform much worse than genre classification.

As long as the predicted trending year is within +/- 5 years of the actual trending year, the prediction will be deemed to be accurate. Roughly 30% to 50% of the results are expected to fall within this range with the CNN, which is much better than roughly 16.6% of the results that would fit into that range if the predictions were made at random by selection a year between 1960 and 2020. However, this is expected to be much lower than the accuracy of genre classification, which on the GTZAN dataset, is shown below in Figure 20, using both a CNN approach and classical machine learning approaches. The gap between the performance of the CNN and classical approaches is expected to widen when predicting the trending year of a song compared to genre classification, due to wider variety of music in a given trending year and the CNN's ability to better capture that wider variety of features compared to classical models. The classical models such as decision trees and random forests are expected to provide results that are roughly 20% accurate because of the inflexibility of these models to handle such a wide variety of data with high noise. Therefore, they would perform only slightly better than a random selection of year. Linear regression and support vector regression is expected to be roughly 25% accurate, which is better than the decision tree and random forest, but still worse than the CNN. The linear regression and SVR models are expected to perform slightly better than decision trees and random forests, based on the fact that in the experiment for genre classification, logistic regression and SVMs performed slightly better than decision trees and random forests as shown in Figure 9.

Model	Accuracy
CNN 2D	0.8320
Decision Tree	0.5160
Random Forest	0.6760
ElasticNet	0.6880
Logistic Regression	0.7640
SVM (RBF)	0.7880

Figure 9: Accuracy of Genre Classifier Models (Guimaraes 2020)

Implementation Details

The CNN implementation was heavily based on prior work in the music-genre-classifcation repository (Cetin 2019), but modified to accommodate for performing a regression on the years versus classification on genres. As described previously, the model consists of 4 convolution layers, 4 pooling layers and 3 linear layers (combined with a ReLU function) at end, which is unlike the genre classification neural network used in the original repository that used a Softmax terminal node. Training is performed using an L1 loss function (unlike genre classification which used cross-entropy), and is trained for 100 epochs on batch sizes of 16 which allows for improved training time (as the optimizer could be invoked after each batch).

An RMSprop optimizer was used, which is an optimizer based on Stochastic Gradient Descent allowing for updating the weights of the model automatically. The optimizer was stepped on each iteration of batch training (which updates the parameters), and had its gradients reset before each iteration of batch training (since by default gradients are accumulated inside buffers when the backwards pass computation is invoked).

The classical models are based on the music genre classification implementation in the GTZAN repository (Guimaraes 2020). However, there were significant changes that were required to make it work for our research question. Each of the models had to be changed from classification to regression. This means changing the logistic regressor to a linear regressor, decision tree classifier to a decision tree regressor, random forest classifier to a random forest regressor, and support vector machine to a support vector regressor. The hyperparameter types and values had to be changed as well to accommodate for the new models.

For the linear regression, the penalty, C value, and max iteration were taken out since they do not apply to linear regression. Fit intercept was set to true so that the line passes through the origin. For the decision tree regressor, the splitter was chosen from best or random based on the output of GridSearchCV. The criterion hyperparameter was changed from ”gini” and ”entropy” to MAE and MSE. The number of estimators and max depth were selected by GridSearchCV from 100, 250, 500, 1000, and 3, 4 respectively. The depths were reduced compared to the original model to prevent overfitting. Also, the criterion was changed to MSE and MAE over ”gini” and ”entropy”. Finally, for the SVR, the C values were changed from $0.5, 1, 2, 5$ to $1, 2, 5, 10, 100$. This is because the SVR was particularly sensitive to hyperpa-

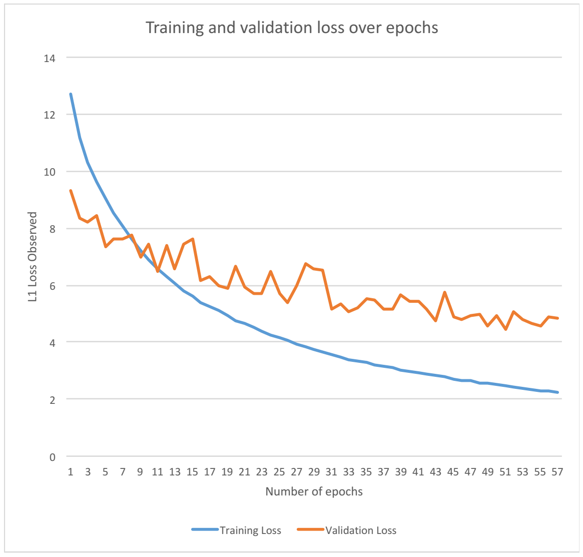

Figure 10: Training and validation loss over epochs. Over time, the CNN improved its validation accuracy, but became slightly overfit as the number of epochs increased. To confirm that the model was not significantly overfit, we tested with an additionally curated dataset of the next 3 (11th to 13th) most popular songs from 1960-2020 from the same billboard and we received similar results as our original benchmarks of MAE of roughly 7.60.

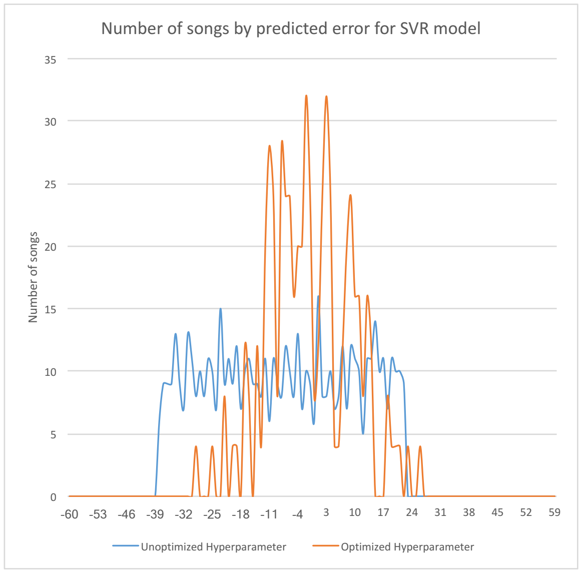

Figure 11: Number of Songs by Predicted Error for SVR Model. The unoptimized SVR model produced very bad results, shown by the blue line. The prediction always showed 1980 +/- 1, due to the high gamma value used while training the model. Allowing GridSearchCV to find lower gamma values allowed the model to perform much better, as shown by the orange line.

rameter values, and a wider range to select from with GridSearchCV allowed for a better model. The ”linear” and ”sigmoid” kernel types were taken out, and instead just the ”rbf” kernel type was used. This is because we have many features and we need multiple dimensions to represent the data. In addition to all of this, the gamma value was selected from 1e-8 to 1e-1 using GridSearchCV. With the default gamma value of 1, the output was limited to very few values and the model performed poorly (see Figure 11).

All of the scoring formulas were changed from an accuracy percentage to the L1 (MAE) function. This is because the problem is no longer a classification problem, and an accuracy percentage does not make sense for a regression problem.

Data Exploration

The features extracted from the songs for the classical models were visualized with LDA, PCA, and TSNE (see Figures 10, 11, 12). The Linear Discriminant Analysis (LDA) visualization is used for showing linear combinations of features that separates classes of objects. The Principal Component Analysis (PCA) is used for emphasizing variation and bringing out strong patterns in the data. The T-Distributed Stochastic Neighbor Embedding (TSNE) model is used for visualization of high dimensional data.

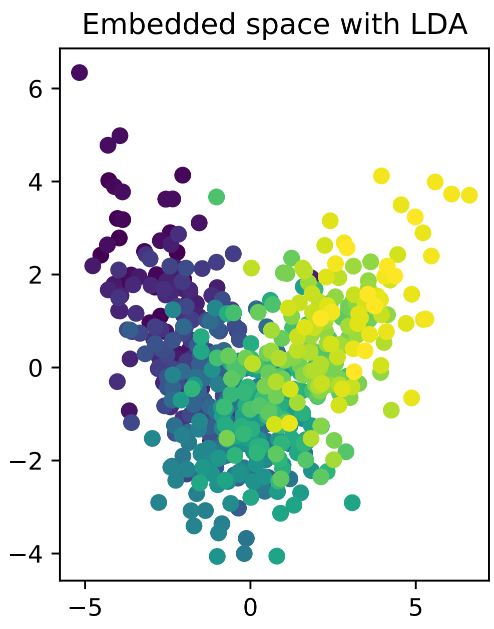

Figure 12: Embedded Space with LDA. This graph shows a V shape in the data. 0 in the point color represents the year 1960 and 60 represents the year 2020. It is clear to see that there is a shift in songs from the year 1960 to 2020.

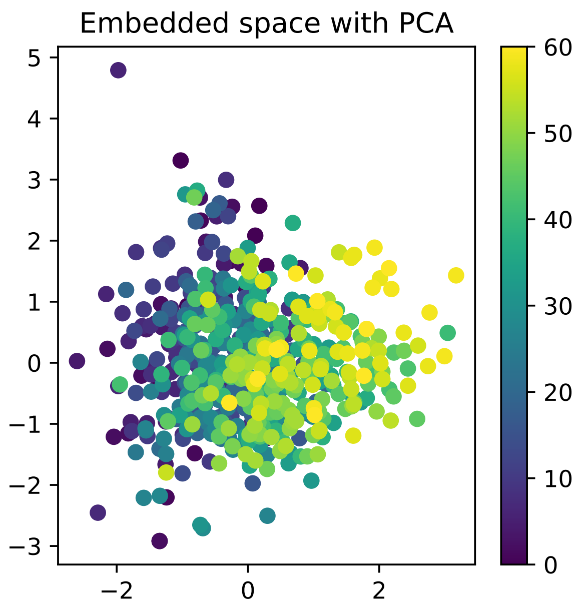

Figure 13: Embedded Space with PCA. This graph shows no strong patterns in the data. 0 in the point color represents the year 1960 and 60 represents the year 2020.

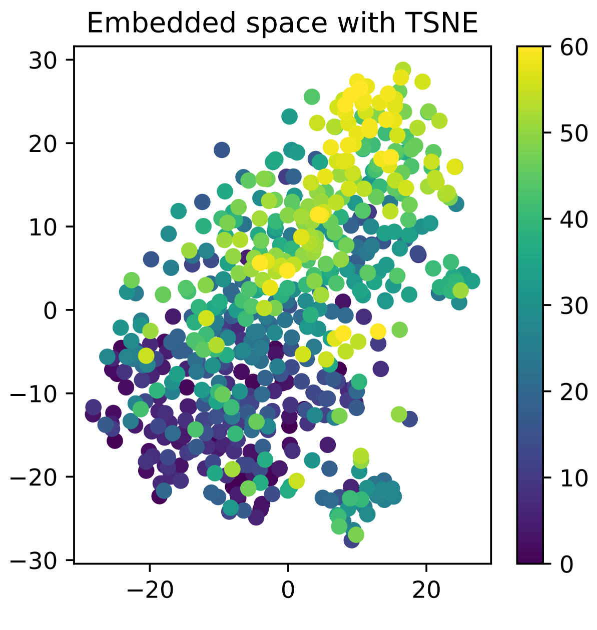

Figure 14: Embedded Space with TSNE. This graph shows a slight pattern in the data, trending upwards. 0 in the point color represents the year 1960 and 60 represents the year 2020.

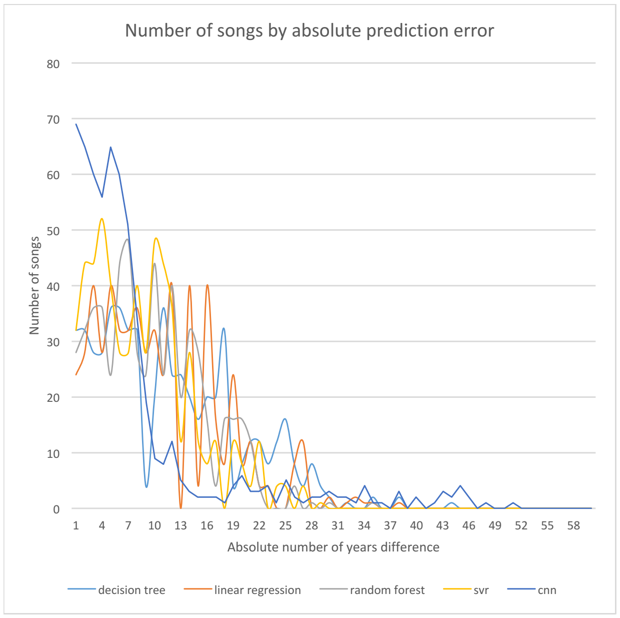

Figure 15: The number of songs plotted against absolute prediction error, which is defined as the absolute value of the predicted song minus the actual release year of the song. It can be observed that CNNs are typically better performing than classical models because it is more positively skewed, and has fewer outliers on the right.

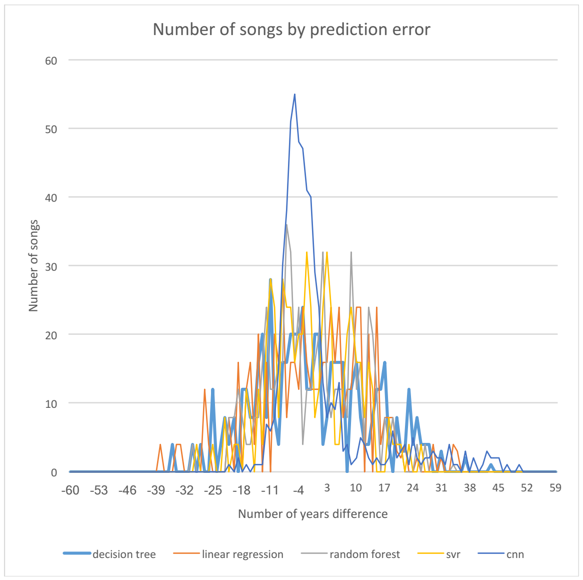

Figure 16: The number of songs plotted against prediction error, which is defined as the value of the predicted song minus the actual release year of the song. It can be observed that CNNs are typically better performing than classical models because it has stronger central tendencies, and is leptokurtic.

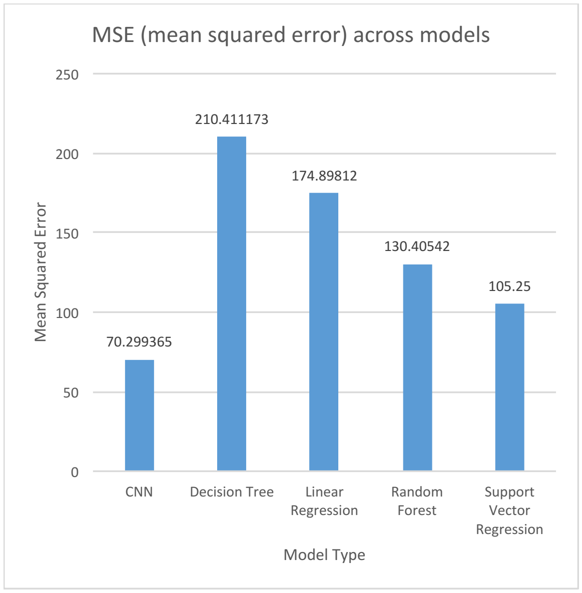

Figure 17: The MSE (mean squared error) across models, which represents the mean squared difference between predicted year and actual release year of songs in the test set. Unlike MAE, the MSE is more sensitive to outliers and therefore a better measure for which model produces more consistently close results. CNN is by far the most accurate, as expected, with decision tree being the least performant.

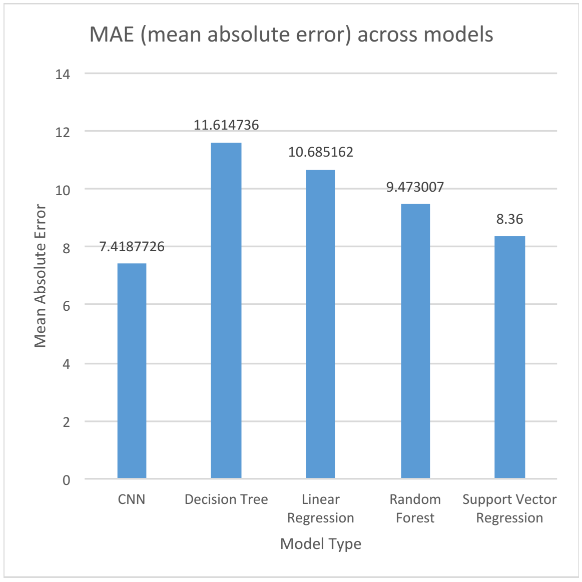

Figure 18: The MAE (mean absolute error) across models, which represents the mean absolute difference between predicted year and actual release year of songs in the test set. The best performing model is the CNN, which is expected, followed by the SVR, with the decision tree performing the worst.

Model	5Y Count	10Y Count	15Y Count
Linear Regression	27.21%	54.42%	72.79%
Decision Tree	26.53%	47.62%	68.03%
Random Forest	26.53%	58.50%	82.99%
SVR	36.05%	65.31%	87.76%
CNN	53.57%	82.82%	87.93%

Figure 19: Accuracy of music trending year prediction models.

Performance of Algorithms

As expected, the CNN model performed much better than all of the other models. This was expected because the CNN model would likely be better at capturing minute differences in songs in each release year compared to the hand picked features for the classical models. This is especially true for music release year prediction, which contains very noisy data due to the wide variety of sounds and genres within each year.

The MAE value over all of the songs for the CNN was 7.2 years, which means that on average, the guess was 7.2 years away from the true value. Given that we tested over a range of 60 years, a prediction of $(2020 - 1960)/2 = 1990$ would have resulted in an MAE value of 15 years, which is the minimum MAE value without knowing anything about the songs. Thus, the CNN performed much better than an ideal prediction without any knowledge of the songs, and much better than a random guess.

The other classical models performed worse than the CNN, but still better than the ideal guess of 1990 (which would minimize the L1 function with no knowledge of the actual songs, given a uniform distribution which we have). Linear regression had an MAE value of 10.68 years, decision trees had an MAE value of 11.6 years, random forests had a MAE value of 9.47 years, and support vector regression had an MAE value of 8.36. All of these results are shown in Figure 18. All of these are higher than the 7.2 years given by the CNN model, but lower than the ideal value of 1990 that minimizes the L1 function with no knowledge of the songs. All of the models performed significantly better than a random guess of the prediction year, which would give an MAE value of roughly 23 years.

The CNN's results are very promising, because a guess within 7 years is what we expect a human to predict, given a random song. An MAE of 7 years means that the prediction will most likely land in the correct decade of the song's actual release, which is quite a promising result. The classical models all performed quite well as well. The decision tree performed the worst, just like in the music genre classification problem. Then, the linear regression came next, followed by the random tree and SVR. This result makes sense because the simple linear regression and decision tree models most likely could not make sense of the many features that were extracted from the songs, due to being simpler models. The random forest model and SVR performed much better, due to being able to better model the complex and noisy data. As expected, the CNN performed the best, since it was able to capture details about the songs using a melspectogram that none of the classical models could capture using the hand-selected features.

The 5, 10, and 15 year proportions are shown in Figure 19. These numbers represent the proportion of predictions that fall within 5, 10, and 15 years of the actual release year of the song. The linear regression, decision tree, and random forest were able to guess within 5 years of the actual value roughly 26% of the time. The SVR was able to do this 36% of the time. The best result was the CNN, which predicted within 5 years 54% of the time. The predictions get much better if we allow the correct range to be within 10 or 15 years, with the vast majority of predictions falling within this range for all of the models.

Lessons Learned

One of the lessons we learned was that adapting similar models to fit different problems can be difficult, as different problems have vastly different ways of using similar models. For example, finding the right features to extract and the right parameters to use can be very challenging. In our case, modifying the music genre classification code bases required changing the models from classification models to regression models, which introduced a variety of parameters we were not familiar with. As a result, the parameters had to be tweaked to fit our new problem, taking up a substantial amount of time in the process. In addition, there was no easy way to determine how good or bad our parameters and feature selections were. The prediction of songs by year is a novel problem as no prior investigation on this topic has been done publicly nor are there any open source implementations to compare to. While we obtained MAE values that appear reasonable, since there has been no other research done on the specific problem of predicting release years of songs, we are uncertain if we reached the true potential of the models.

One particular example of a problem that we faced was working with the support vector regressor (SVR). All of the prediction values were within one year of 190 when the gamma value was set to 1. After further investigation, it was determined that the model was very sensitive to the gamma value. This called for the use of GridSearchCV in the SKLearn library to determine the best value of gamma. The choices that we selected were $1e-9$, $1e-8$, ... ,$1e-1$. As a result, we built a model that predicted the release year of a song with far greater precision (See Figure 11).

We did not think that machine learning models would be so sensitive to these tuning parameters, but we were proven wrong when an incorrect value for these parameters would lead to inept and unavailing results. There were a lot of different parameters to learn about, and there are still lengths to go to fully understand each of these parameters, within both the classical models and CNN's. However, throughout the process, we still learned a lot about linear regression, decision trees, random forests, support vector regression, and CNNs. The biggest takeaway was how each model was highly sensitive to variations in its hyperparameters.

Our repository [1] is open sourced on GitHub.

[1]https://github.com/peterhuangSC/music-year-predictor

Discussion

Interpretation

In line with our initial hypothesis, the CNN model performed much better than the classical machine learning models such as linear regression, support vector regression, decision tree, and random forest. This is likely because the CNN model along with the spectrograms were better able to capture details in the dataset that hand selected features could not. The hand selected features are features that may not necessarily be good for the models, but have been somewhat arbitrarily picked. A spectrogram, by contrast, examines the musical data as a whole and can capture patterns and details that would not be visible with hand selected features.

In addition to that, a CNN by nature is more complex than any of the classical machine learning models. It is better at capturing multidimensional details compared to simpler models such as linear regression or decision trees. This explains why the results were better for the CNN model when compared to the classical models.

With respect to our research question, it was clear from our experiment that CNNs do a very good job at predicting the trending year of a song, at least with the dataset that we used, taking the top billboard songs per year from 1960 to present. It performed better than expectations, and nearly as well as the CNNs used for genre classification. The simpler models, however, handled music trending year prediction worse than it handled genre classification. This is likely because genres have simpler characteristics that make it easier for simpler models to create an accurate model on. Overall, the results were generally in line with what was expected, and for the CNN model, slightly better than expected.

Implications

The implications of these results is that machine learning and AI can be further utilized by the industry and researchers to achieve things that have not been achieved before. For example, if music trending year prediction is possible, is it possible to also predict where the music industry is heading? If that is true, then the music industry leaders can plan in advance about the direction of their artists and song releases.

Although genre classification has been heavily explored as mentioned in the related works that were surveyed earlier, music trending year prediction is a new branch of study when it comes to machine learning and music. It reinforces the idea of using machine learning and AI to improve music recommendation systems and helping people find the music they like. As with other studies that compare CNNs with classical machine learning models such as linear regression, support vector regression, decision trees, and random forests for music genre classification, the CNN performed the best. This was the case with music genre classification, and it is also the case with music trending year prediction. This is also likely for the same reasons why CNNs worked so well for music genre classification. It is because CNNs, along with mel spectrograms, are able to capture details that human selected features are not able to. CNNs are an incredibly powerful tool for building models on highly complex data, and these results further reinforce that idea.

Limitations

There are, of course, some limitations to this research. Due to the limited size of our dataset (600 songs), we were not able to capture the massive variety of songs available from each year. There are thousands of songs released every year, and our research only used songs from the BillBoard Top 100. This means that the results may potentially only apply to pop songs rather than other genres, because pop songs tend to be on the BillBoard Top 100. This would leave out other, less mainstream songs. Using a different dataset for this project could have led to different, possibly better or worse results.

In addition to that, we used songs from the US BillBoard Top 100. This means that the research may only be relevant to US and Western songs, and not songs from other places around the world. Even though the models worked well for the US BillBoard Top 100 songs, they may not work well for songs from India or China, for example, which are likely fundamentally different in nature.

One other feature that we could have used could have been the lyrics of a song. Lyrics can sometimes give hints as to when a song was trending or released, since lyrics contain cultural information that can pertain to certain time periods. For example, if a song lyric mentions ”phone”, the models would likely know that it is a fairly recent song. However, including this feature would have highly complicated our work, since we would have to build two orthogonal models, and somehow combine the results. For example, we would need to build the models like we have done with musical characteristics and mel spectrograms, and we would also need to build models based purely on the lyrics. In addition, lyrics may not necessarily provide the results that we would want, which is to determine the sound characteristic of a year, rather than the words that are spoken.

Conclusion

The problem that we set out to solve was determining how machine learning and AI techniques such as CNNs, linear regression, support vector regression, decision trees, and random forests would perform when predicting the release year of a song. In particular, we wanted to compare the performance between the CNN model and the classical models. The motivation behind this was to explore an area where music recommendation algorithms can be improved, so that music streaming applications such as Spotify and Apple Music can recommend songs that are more tailored to each individual. With this new approach to predicting the trending year of a song, music recommendation algorithms will be able to suggest a larger category of music than it previously could. Someone who enjoys listening to a sound signature from a certain time period (e.g. 1960s music), would be able to find music that sounds like it is from the 1960s even if it was released much later.

The results showed that CNNs performed very well when predicting the trending year of songs between 1960 and 2020. It predicted correctly within 5 years of the actual trending year 53% of the time, and within 10 years of the actual trending year 83% of the time. The classical models performed slightly worse, predicting within 5 years 26% to 36% of the time, and within 10 years 47% to 65% of the time. Out of each of the classical models, the support vector regression performed the best, followed by linear regression and random forest. The decision tree performed the worst. These results likely show that the CNN model was able to capture minute details in the songs that the classical models were not able to.

In the future, we would like to see more work done with tuning all of the hyperparameters. Although we have tried our best to tune each of the hyperparameters, it is not clear whether our chosen hyperparameters create models that perform ideally. Since there are no other similar projects to compare the results to, it is difficult to determine whether the models are fully utilized for this regression problem. There could be many areas for optimization that could change these results.

Also, the feature extractions for the classical models can also be modified to better fit the problem. We used the feature extractions that were generally used for genre classification, but there is no guarantee that these features work as well for predicting the trending year of a song. Trying different feature extractions may lead to different results.

It would be interesting to see more related work in this area. Although genre classification has been heavily explored, classification of music through other means has not been explored very much. With further exploration in this area, music streaming applications will be able to better recommend music to match peoples' music tastes.

Hyperparameter	Specific Values	Justification
Convolution Layer 1	In channels = 1 Kernel size = 3 Stride = 1 Padding = 1 Out channels = 64	Follows the parent model's choice for genre classification.
Convolution Layer 2	In channels = 64 Kernel size = 3 Stride = 1 Padding = 1 Out channels = 128	Follows the parent model's choice for genre classification, and has in channels corresponding to the out channels of conv1.
Convolution Layer 3	In channels = 64 Kernel size = 3 Stride = 1 Padding = 1 Out channels = 128	Follows the parent model's choice for genre classification, and has in channels corresponding to the out channels of conv2.
Convolution Layer 4	In channels = 256 Kernel size = 3 Stride = 1 Padding = 1 Out channels = 512	Follows the parent model's choice for genre classification, and has in channels corresponding to the out channels of conv3.
Fully Connected Layer 1 (linear transformation)	In features = 2048 Out features = 1024	Follows the parent model's choice for genre classification.
Fully Connected Layer 2 (linear transformation)	In features = 1024 Out features = 256	Follows the parent model's choice for genre classification, and has in features corresponding to the out features of fc1.
Fully Connected Layer 3 (ReLU)	In features = 256 Out features = 1	Follows the parent model's choice for genre classification, and has in features corresponding to the out features of fc2. Out features need to be equal to 1, which will be the float indicating the predicted year.
Pool Layer 1	Kernel size = 2	Follows the parent model's choice for genre classification.
Pool Layer 2	Kernel size = 2	Follows the parent model's choice for genre classification.
Pool Layer 3	Kernel size = 4	Follows the parent model's choice for genre classification.
Pool Layer 4	Kernel size = 4	Follows the parent model's choice for genre classification.
Epoch number	250	Follows the parent model's choice for genre classification.
Batch size	16	Follows the parent model's choice for genre classification.
Criterion	Type = Mean Absolute Error Loss	The Mean Absolute Error is one of the most commonly used function for regression.

Figure 20: CNN Parameters and Values, based on the parent model (github.com/cetinsamet/music-genre-classification) which received over 70% test accuracy on the GTZAN dataset for genre classification are used as the initial values.

Parameter	Specific Values	Justification
Criterion	”mse”, ”mae”	These are all the possible values for the criterion.
Splitter	”best”, ”random”	Follows the parent model’s choice for genre classification.
Max Depth	”3”, ”4”	These max depth values produced the best results.

Figure 21: Decision Tree Parameters and Values

Parameter	Specific Values	Justification
Number of Estimators	100, 250, 500, 1000	Follows the parent model’s choice for genre classification.
Criterion	”mse”, ”mae”	These are all the possible values for the criterion.
Max Depth	5, 7, or None	These values gave the best results.

Figure 22: Random Forest Parameters and Values

Parameter	Specific Values	Justification
Fit Intercept	True	Fitting the intercept forces the data to cross the origin.

Figure 23: Linear Regression Parameters and Values

Parameter	Specific Values	Justification
Kernel	”rbf”	This value gave better results that ”linear”.
C	1, 2, 5, 10, 100	These values gave the best results.
Epsilon	0.0001, 0.001, 0.01, 0.1, 10, 50	These values gave the best results.
Gamma	”auto”, ”scale”, 1e-8, 1e-7, ... 1e-1	These values allow for a greater degree of predictions compared to the default of 1.

Figure 24: SVR Parameters and Values

References

Bhattacharyya, I. 2018. Support vector regression or svr. Available at https://medium.com/coinmonks/support-vector-regression-or-svr-8eb3acf6d0ff.

BillboardTop100. 2020. Billboard top 100. Available at http://billboardtop100of.com/.

Cetin, S. 2019. Music genre classification. Available at https://github.com/cetinsamet/music-genre-classification.

Chakure, A. 2019. Random forest regression. Available at https://medium.com/@aaaanchakure/random-forest-and-its-implementation-71824ced454f.

Choi, K.; Fazekas, G.; Sandler, M.; and Cho, K. 2018. A comparison of audio signal preprocessing methods for deep neural networks on music tagging.

Dannenberg, R. B.; Thom, B.; and Watson, D. 1997. A machine learning approach to musical style recognition.

Google. 2020. Multi-class neural networks: Softmax.

Goulart, A. J. H.; Guido, R. C.; and Maciel, C. D. 2012. Exploring different approaches for music genre classification.

Guimaraes, H. 2020. gztan_keras. Available at https://github.com/Hguimaraes/gtzan.keras.

ICLR. 2018. How do deep convolutional neural networks learn from raw audio waveforms?

Jiang, D.-N.; Lu, L.; Zhang, H.-J.; Tao, J.-H.; and Cai, L.-H. 2002. Music type classification by spectral contrast feature.

LeCun, Y.; Haffner, P.; Bottou, L.; and Bengio, Y. 1999. Object recognition with gradient-based learning.

Li, T.; Ogihara, M.; and Li, Q. 2003. A comparative study on content-based music genre classification.

Li, H.; Xue, S.; and Zhang, J. 2018. Combining cnn and classical algorithms for music genre classification.

McKay, C., and Fujinaga, I. 2006. Musical genre classification: Is it worth pursuing and how can it be improved?

Ng, A. 2017. Improving deep neural networks: Hyperparameter tuning, regularization and optimization.

Russell, S. J., and Norvig, P. 2016. *Artificial intelligence: a modern approach*. Upper Saddle River, NJ, USA; Pearson Education Limited,.

Shafkat, I. 2018. Intuitively understanding convolutions for deep learning. Available at https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1.

Sturm, B. L. 2013. The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use.

Tyagi, N. 2020. A brief guide to recreational pyromania. Available at https://www.analyticssteps.com/blogs/how-spotify-uses-machine-learning-models (2020/09/25).

Tzanetakis, G.; Essl, G.; and Cook, P. 2002. Automatic musical genre classification of audio signals.

Zhang, W.; Lei, W.; Xu, X.; and Xing, X. 2016. Improved music genre classification with convolutional neural networks.