Deep Learning and NLP Final Project - Genre Recognition

- Lior Breitman 212733257
- Talia Seada 211551601
- Netanel Levine 312512619

  https://github.com/LiorBreitman8234/genreDetectionProject

Abstract:

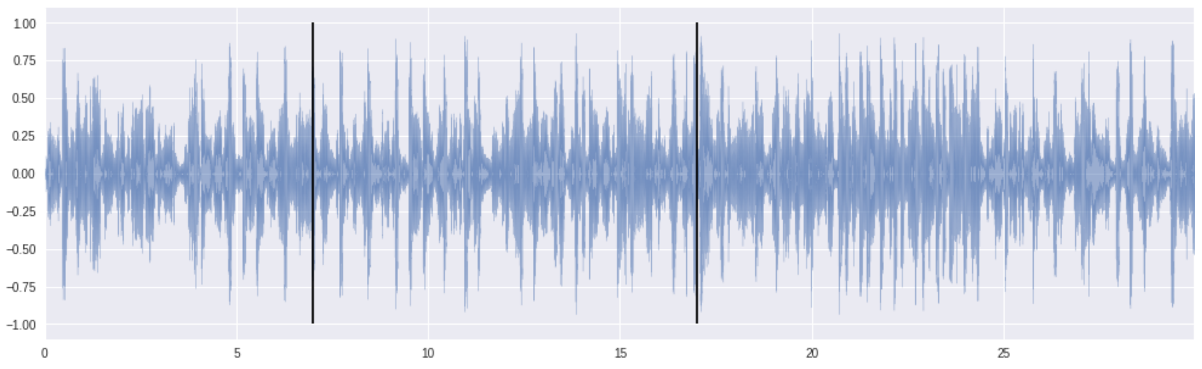
Predicting Song Genres using Convolutional Neural Networks.
In this study, we present a method for predicting the genre of a given song using convolutional neural networks (CNNs). Music genre classification is a challenging task due to the complex and subjective nature of music. However, it is an important problem with many practical applications, such as music recommendation systems and playlist generation. To address this problem, trained a CNN model on a dataset of songs labeled with their corresponding genres. Our model achieved an impressive accuracy of 86% on the test set, demonstrating the effectiveness of CNNs for music genre classification.

Introduction:

For this project, we sought out data that was not overly simple and that could be processed in multiple ways. We chose to work with music data because it can be processed directly from the audio files and also using features extracted from the audio. After searching for datasets, we came across the FMA repository which contained a large amount of song data in the form of both audio and extracted features such as chromagram and spectral centroid. While we did not have time to train on raw audio, we were able to achieve good results using only the extracted features.

Similar Projects:

1. https://github.com/mdeff/fma
2. https://github.com/maximilian-witt/music_classification
3. https://github.com/nahue-passano/music_genre_classification



Our Data:

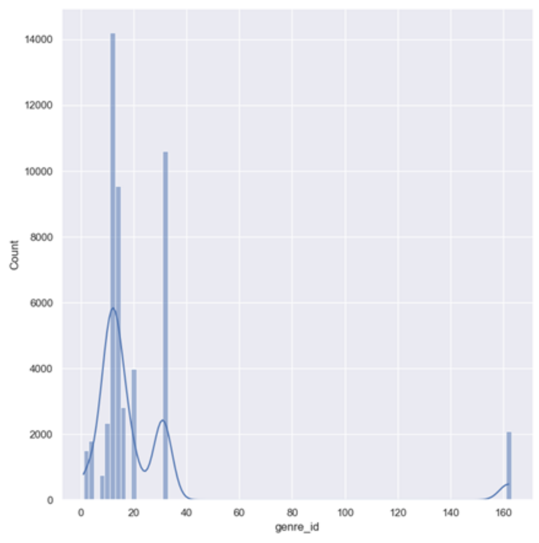
The FMA repository contains a large dataset of songs in both audio form and with extracted features such as chromagram and spectral centroid. We initially planned to work with the raw audio data, but due to time constraints, we used the features provided in a CSV file instead.

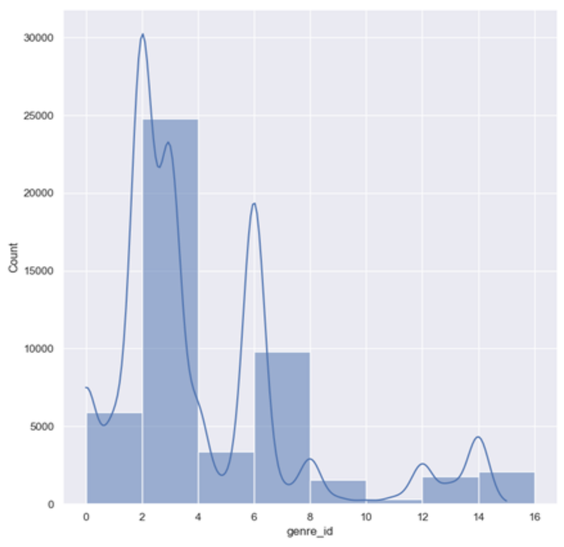
The dataset included 518 features (columns) and 161 classes (genres). We added the genre label for each song using the tracks.csv file, and only used songs with a pre-defined genre. This resulted in a training set of around 40,000 songs. To ensure the model had sufficient data to learn from, we allocated 80% of the data to the training set and split the remaining 20% equally between the validation and test sets.

Preprocessing:

During the data preparation phase, it was observed that there existed a significant imbalance in the distribution of samples among the classes, with certain classes having a relatively low number of samples compared to others. To address this issue, the initial step taken was to remove all classes that did not have any samples associated with them, resulting in a reduced number of classes from one hundred and sixty one to sixteen.

Subsequently, an analysis of the sample distribution revealed that three classes had more than nine thousand samples, while one class had only twenty four samples. To address this imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) was employed to generate additional samples for the under-represented classes, with the goal of achieving a more balanced distribution of samples among the classes.





Our Projects:

Baseline Models:

The FMA repository provided several basic models that they trained on the data, and we could use them as a benchmark for our models. The highest accuracy they achieved was 62.88% using a support vector machine (SVM). They also ran logistic regression and multi-layer perceptron (MLP) models, achieving accuracies of 61.1% and 58.14% and 57.95%, respectively. The MLP models had one hidden layer with 100 perceptrons and two hidden layers with 200 and 50 perceptrons, respectively. These results served as a baseline for comparison to the models we developed in the study.

Basic Models:

In the first part of the project, we developed a logistic regression model using Softmax and simple MLP models with 3 and 4 hidden layers. The logistic regression model achieved the highest accuracy of 63.6% on the test set, while the MLP models achieved accuracies of 58.4% and 59.7%, respectively. These results were slightly better than the benchmark models provided in the FMA repository, which may be due to the use of TensorFlow tools rather than scikit-learn. We also observed that the logistic regression model had a higher accuracy compared to the MLP models. This raises questions for the next part of the project when we consider more complex techniques such as CNNs and RNNs for classification. It may be more effective to use a simpler model that has a good accuracy while also requiring less time to train.

Complex Models:

Based on the fact that the data used had already undergone feature extraction, we determined that a CNN model would be the most appropriate approach. As there was no benchmark provided in the FMA repository, we evaluated the performance of our CNN model against previous models we had built.
To achieve optimal results, we experimented with various configurations of layer order, size, and quantity. Our final CNN model includes a combination of:

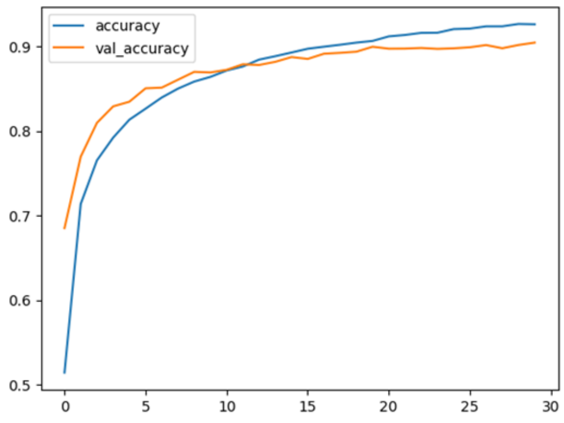
- Convolution layers with max pooling
- Dropout layers for regularization
- A Flatten layer to transition to the fully connected portion of the network
- Fully connected dense layers with regularization
- Softmax activation function in the last layer to accommodate the 16 classes present in the data.

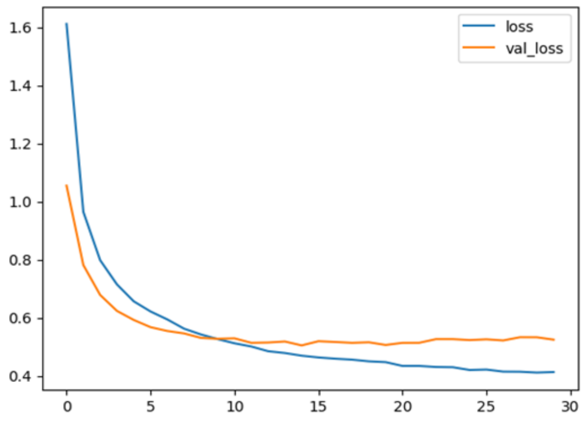
Conclusion:

We found that using deep learning methods yielded better results than using traditional machine learning methods for this complex problem. During the fine-tuning process, we experimented with altering the network structure by adding or removing layers. We observed that the imbalanced nature of the data greatly impacted the results. To address this, we attempted to balance the data and found that it significantly improved the accuracy of our networks. It is possible that with more data for under-represented genres, the models would perform even better without the need for manual data balancing.

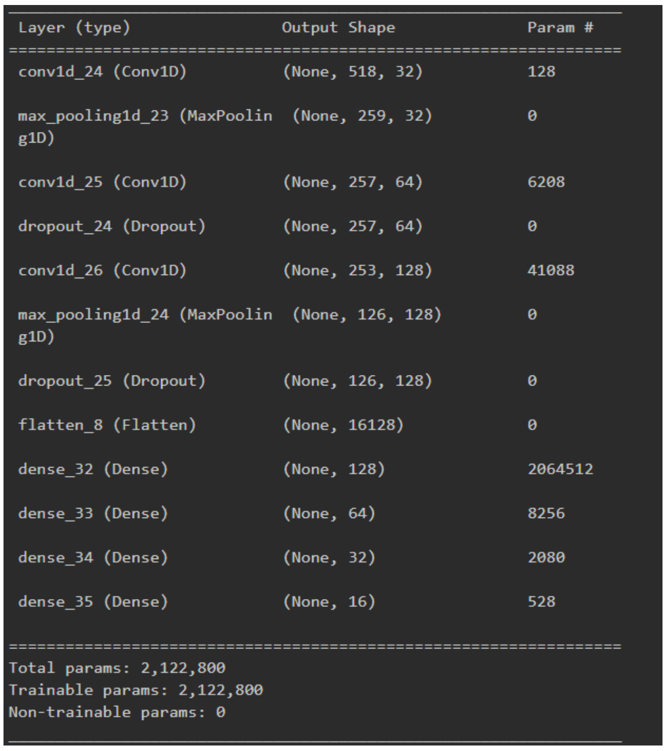
Results:

The model that performed the best on the test set achieved a success rate of 89.5%. It was evident that by balancing the data and implementing regularization techniques, the model's performance was improved.





The structure of the model:



```
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_24 (Conv1D)          (None, 518, 32)           128

 max_pooling1d_23 (MaxPoolin  (None, 259, 32)          0
 g1D)

 conv1d_25 (Conv1D)          (None, 257, 64)           6208

 dropout_24 (Dropout)        (None, 257, 64)           0

 conv1d_26 (Conv1D)          (None, 253, 128)          41088

 max_pooling1d_24 (MaxPoolin  (None, 126, 128)         0
 g1D)

 dropout_25 (Dropout)        (None, 126, 128)          0

 flatten_8 (Flatten)         (None, 16128)             0

 dense_32 (Dense)            (None, 128)               2064512

 dense_33 (Dense)            (None, 64)                8256

 dense_34 (Dense)            (None, 32)                2080

 dense_35 (Dense)            (None, 16)                528

=================================================================
Total params: 2,122,800
Trainable params: 2,122,800
Non-trainable params: 0
_________________________________________________________________
```