Music Genre Classification Based On Audio

Horváth Szilárd, Czumbel Péter

1. Introduction

Music genres have different characteristics and unique features. Humans are good at differentiating between genres, because we are inherently good at recognizing features that define each genre like certain instruments, the rhythm or pace. Machine learning models on the other hand have a more challenging task labeling music based on genre, because most of the time they only have the audio files as data.They have to find the relevant patterns and sequences in the data, they have to find what parts and features do humans sense more than others.

As machine learning algorithms became more and more advanced it became possible to recognize these features not only in music but natural speech as well to a degree. In the future this part of machine learning could be a crucial part in the communication between humans and machines.

In this project we try different deep learning techniques for music genre classification and compare the results.

2. Dataset

We are using the GTZAN dataset, which consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050 Hz Mono 16-bit audio files in .wav format. It includes the following genres: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock. We thought some of these genres should be easily differentiable like rock and classical, and some of them would be harder like pop and jazz.

3. Previous Solutions

Some of the best performing models for music genre classification use language processing models. This needs a specific type of data that can be converted to text and tokenized. Most models we found with this approach used midi files.

For the Gtzan dataset we found interesting and diverse solutions. Right now the best performing model on kaggle is M2D which is based on masked autoencoding the audio. It uses two networks, one online network that predicts the masked data and another network to validate it.

We also found models which use the Mel spectrogram of the audio so they turn the task into image related machine learning.

4. Proposed Solution

4.1 Data preparation

Because the dataset doesn’t have that much data, we decided that we will split each track into segments. This also helps the model to generalize more, because it learns shorter patterns, which are more defined by genre, rather than the whole track, which depends on the individual song more than on the genre.

We also used the audio’s Mel Frequency Cepstral Coefficients. It is an audio transformation technique which focuses on mimicking the human perception of sound. In the case of music, the genres are decided by humans, that is why we decided for this transformation. It extracts several features from each segment.

For the CNN we decided to use the mel spectrogram of the audio, it became an image classification task. For this we extracted 128 features from MFCC in this case and separated each audio track into 15 segments.

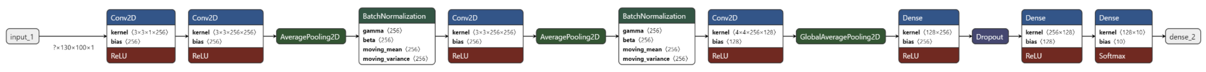
We decided to split each 30s track into ten 3s long segments. After that we extracted 13 features from their MFCC. We labeled each segment with its track’s genre. We split this dataset into Train, Validation and Test datasets with 70%-20%-10% partitioning.

4.2 Models

We tried different kinds of models for the task. We focused on models, which excel at sequential data, because in the case of music the sequence and the surroundings of the data are very important. We used three different approaches to processing these sequences, but all of them ended up with roughly similar results.

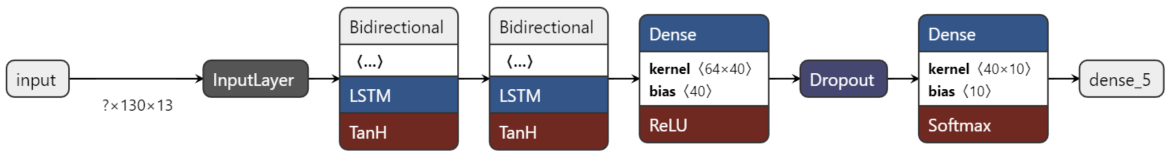
4.2.1 CNN

For the CNN we decided to use the spectrograms instead of the Mel coefficients. We used Batch Normalization, AveragePooling and Dropout layers to increase the accuracy and decrease the chance of overfitting. We tried out different activation functions and initializers for the layers, but in the end we settled for relu as activation function. Our model is completely sequential, as our experiments to add skip and residual connections did not improve the validation accuracy of the model.



4.2.2 LSTM

LSTMs primarily use sequential data, so it was an obvious choice for this task. We stacked multiple LSTMs vertically so it can learn more complex patterns. We tried out different sizes for the LSTMs. We also tried to add dropout layers between the LSTM layers, but it decreased the accuracy of the model, so we decided that we only put dropout after the LSTMs in the dense layers. We tried using bidirectional LSTM layers, but the accuracy did not improve significantly over our regular LSTM model, and both versions performed worse than the convolutional model.



4.2.3 Transformer

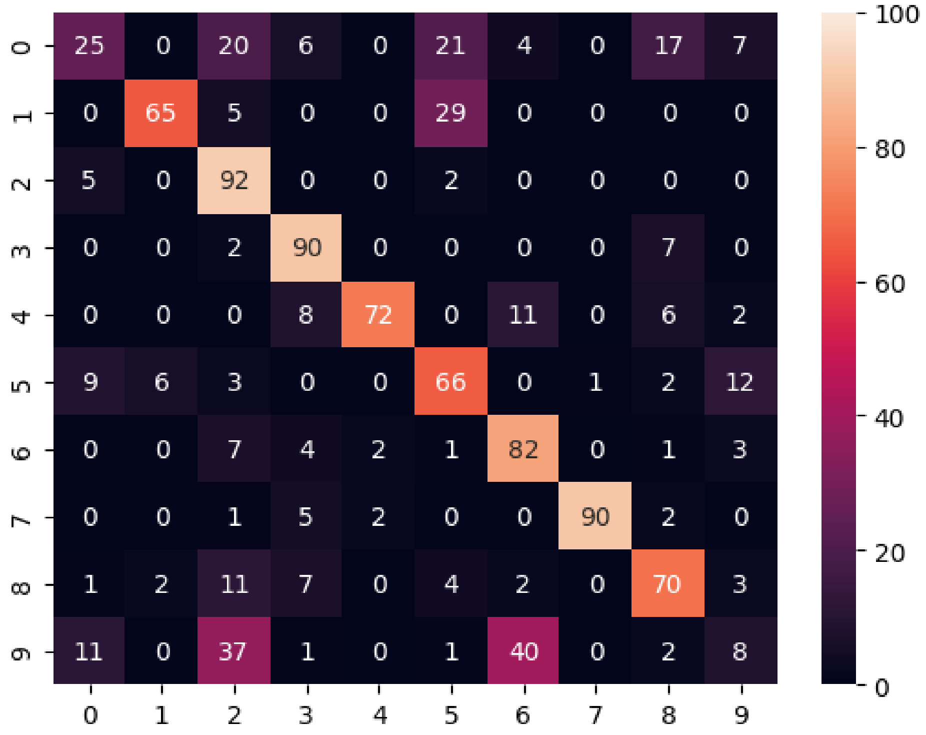
We chose to try out a transformer model because transformers have shown great effectiveness in processing sequential data, such as natural language and audio signals. Music is essentially a sequential data format, where the order of notes and rhythms contributes to its genre. Transformers excel in capturing long-range dependencies in sequences, making them suitable for analyzing complex musical patterns and recognizing genre-specific characteristics. Additionally, transformer models can learn hierarchical representations of data, allowing them to capture both local and global features that contribute to genre classification.

In natural language processing, where transformers are most used, the models work on easily tokenizable data, however when processing audio, that's a much harder problem. Some previous solutions utilizing transformers circumvented this problem by training the model on sheet music, or midi format, so the data used for training was already in a tokenized format.

To avoid having to tokenize our data, we used an encoder based approach, where we sequentially ran the input through multiple encoder layers with a self attention mechanism, which is basically the encoder part of an autoencoder with added multi headed attention layers. We simply connected the output of the encoder to two dense layers. This model achieved an accuracy slightly worse than the convolutional model, but better than the LSTM.

5. Evaluation

Music genre classification is a classification task, so we can use the training data to make the models predict their output than compare it to the actual output. For this purpose we used Sickit-learn's classification report and for visualization we used heatmaps. The heatmap below is the confusion matrix of our best performing convolutional network.



The two genres that the model had the most problems with were blues and reggae, those two were often confused for rock and pop. This model achieved an overall accuracy of 66%, other scores and metrics are in the table below.

```
              precision    recall  f1-score   support

           0       0.49      0.25      0.33       100
           1       0.89      0.66      0.76        99
           2       0.52      0.93      0.66        99
           3       0.74      0.91      0.82        99
           4       0.95      0.73      0.82        99
           5       0.53      0.67      0.59        99
           6       0.59      0.82      0.69       100
           7       0.99      0.90      0.94       100
           8       0.65      0.70      0.68       100
           9       0.23      0.08      0.12       100

    accuracy                           0.66       995
   macro avg       0.66      0.66      0.64       995
weighted avg       0.66      0.66      0.64       995
```

6. Conclusion

This classification task ended up being harder than we expected. Mostly because of the small size of the dataset, the models very easily overfit, and using tricks to increase the size of the dataset, like splitting up the tracks into pieces, reduces the difference between the different genres, making the classification even harder. The models can only be as good as the dataset. Between the three models the CNN model which was based on the image representation of the data performed the best.