Music Classification

Project report

Camille Lhoir, Boya Zhang

April 30, 2021

Engineering Science

Contents

1 Introduction

It has never been easier to find music thanks to websites or streaming services such as YouTube and Spotify. That is why music classification is more important than ever. The amount of available makes it hard for one person to find what they like and separating it into different categories is a helpful start. However it also means that it is humanly impossible for a person to catalog the entire discography found online. That is where machine learning and deep learning comes into play.

One way to classify music is by genres. [3] It is the most popular one and therefore the most logical for many people. It should be noted however that it is a very subjective task as definitions of genres vary depending on the person, the place or the time. For example, The Beatles were considered a pop band in the 60s but now Taylor Swift is put into the pop category. And actually, ten years ago, she was classified as a country artist. Music has evolved over time and that should definitely be kept in mind as this report goes on. However, this adds a lot more complexity and only standard, non gender-bending data will be considered for this project.

This report will first discuss the state-of-the-art models in section 2. Section 3 will explain in details the models used for this project and the components will be studied in section 4. Finally, the conclusion can be found in section 5.

2 Literature review

When it comes to categorizing any sort of audio signal, the first question concerns the feature extraction. Features are meant to represent the audio signals well and have a definite impact on the model accuracy in the end. The most common ones come from speech recognition and are called Mel-frequency cepstral coefficients (MFCCs). Other common hand engineered features are timbral texture, pitch and rhythm. [8] However, many papers have tried different sets as well (either new ones all together or adding to already defined features): Daubechies Wavelet Coefficient Histograms (DWCH) [4], entropies and fractal dimensions [3]. Experiments showed better results with frequency-based parameters even though they require a longer runtime than time-based parameters. [3] Another approach is to train a CNN model to extract itself the features it deems necessary for music classification. [1, 2]

For classification, different models have been used over the time. A Gaussian Mixture model was used for classifying 10 genres and showed an accuracy of 61%. [8] Support Vector Machines can also be used. In one case, 3 machines were used for 3 genres (each machine is trained to return 1 for its assigned genre) but their experiments only considered 90 extracts which seems too small a database to yield pertinent accuracy results. In another case, SVMs were used pairwise and DWCH with timbral features (MFCC and FFT) were used as features. The model yielded the best results: 80% of accuracy. [4] CNN models showed good results as well with one achieving 70% of accuracy on 10 different genres. Another popular neural network structure for this kind of problem is the Long Short-Term Memory network. This kind of network (Recurrent Neural Network) can incorporate previous information stored in the memory, into the prediction. It has been used alongside MFCCs and yielded 53% accuracy in [7].

To train those models, usually labeled data is used. To two main databases are GTZAN Genre Collection and the Million Song Dataset. The former is considerably shorter with only

1000 30 seconds extracts (100 for each genre) and the latter only contains metadata, not the actual audio files. [5]

3 Models implemented

All codes were done using Tensorflow (Keras) and were mainly based on the labs linked to the course and tutorials found on YouTube. [9]

3.1 Preprocessing

For this project, the GTZAN dataset was used. It might be smaller than MSD but it seemed more appropriate as it has the actual data and not just metadata. Therefore the models obtained can be used on real songs, which gives a clearer understanding to the user. All 10 genres are considered in this project which adds some difficulties for the models to differentiate.

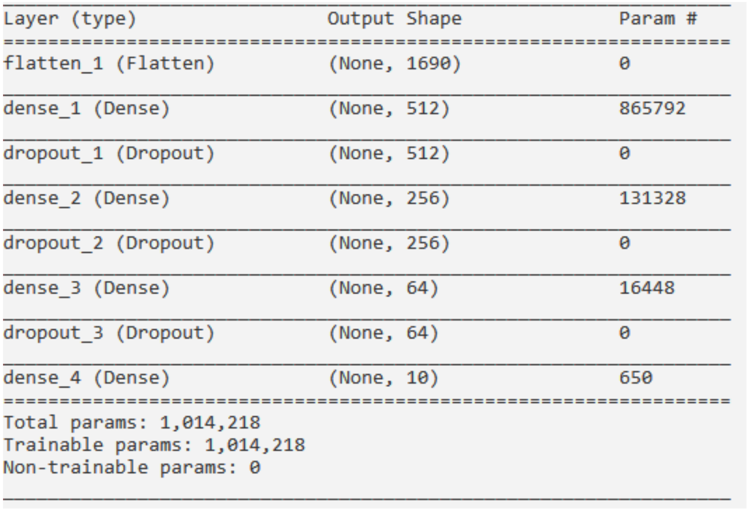
For the features, the MFCCs were chosen due to their documented efficiency in the case of music classification. The Python package librosa also made it easy to extract them. For this project 13 MFCCs were extracted as it is traditionally how many coefficients are needed to represent the audio data. [6]

Another feature was explored: tempo. It is a time domain feature to compare with the MFCCs which are frequency domain features. And it can be extracted using librosa.

3.2 Neural Network

The first model implemented is a simple neural network with 3 hidden fully-connected layers. The inputs are the MFCCs (13 features) mentioned before. As they are in the form of a 2-D matrix, the input layer must flatten them to forward them in the network. The ReLU activation function is used for the hidden layers. The output layer has 10 neurons for the 10 genres and uses the softmax activation function as this is a multiclass classification problem. To avoid overfitting, dropout and regularization have been applied to the hidden layers. The model structure can be seen in Figure 1.

The same structure was used for the tempo feature input.

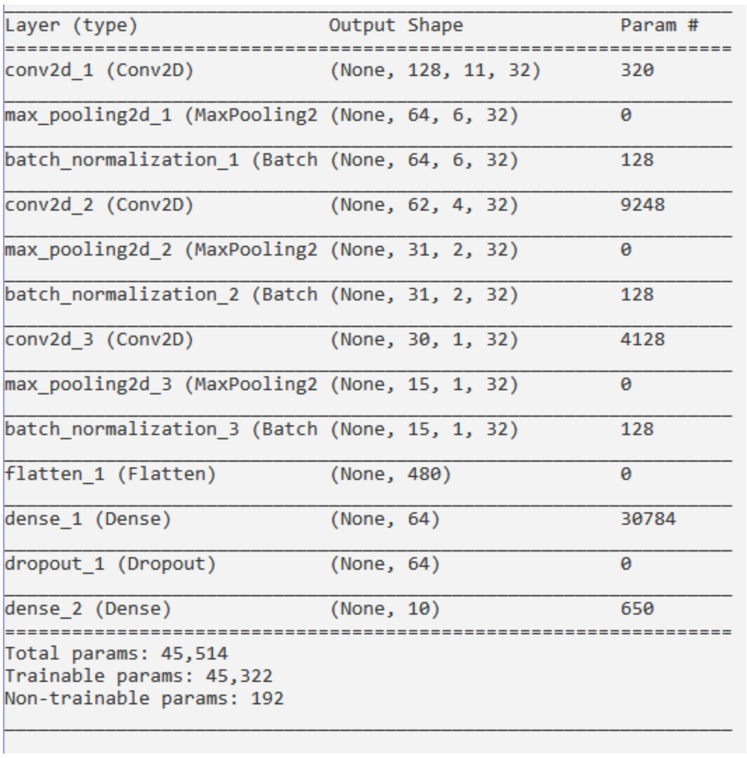


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten_1 (Flatten)          (None, 1690)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               865792    
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328    
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 64)                16448     
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_4 (Dense)              (None, 10)                650       
=================================================================
Total params: 1,014,218
Trainable params: 1,014,218
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Neural Network Structure

3.3 Convolution Neural Network

The second one is a convolution neural network. They are often used for image processing and audio in a way can be considered as an image. The inputs are two dimensional and can be fed into the network as they are (though adding another dimension to meet the keras 2D convolution layers requirements). For this network 3 convolution layers were added. Each has 32 filters and is followed by a MaxPool layer and a BatchNormalization layer to speed up training and get a more reliable layer. Then a fully-connected layer is added with 64 neurons and dropout is applied to avoid overfitting like in the previous model. The output layer is the same as the first model as the prediction is the same format. The model structure can be in Figure 2.

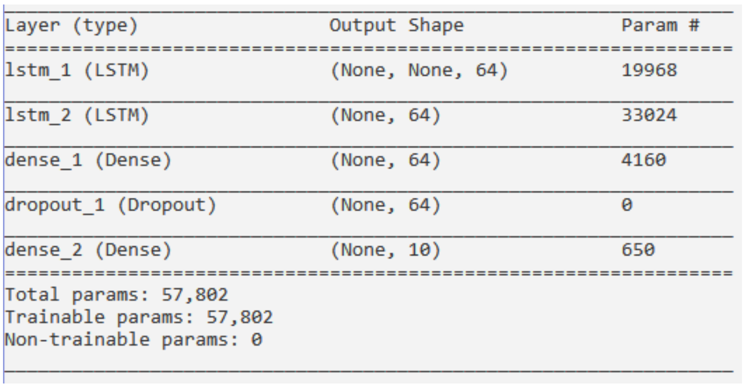


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 128, 11, 32)       320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 6, 32)         0
_________________________________________________________________
batch_normalization_1 (Batch (None, 64, 6, 32)         128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 62, 4, 32)         9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 31, 2, 32)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 31, 2, 32)         128
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 30, 1, 32)         4128
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 15, 1, 32)         0
_________________________________________________________________
batch_normalization_3 (Batch (None, 15, 1, 32)         128
_________________________________________________________________
flatten_1 (Flatten)          (None, 480)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                30784
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 45,514
Trainable params: 45,322
Non-trainable params: 192
_________________________________________________________________
```

Figure 2: CNN Structure

3.4 LSTM Neural Network

Lastly, Recurrent Neural Networks are often used when the data is made up of different sequences that are expected to have some kind of codependency. In machine learning, RNN-LSTM (Long Short-Term Memory) networks are especially useful and therefore were chosen for this project. LSTM have shown better results because it allows "a longer memory" than RNN. This network contains less layers than the previous one, only 2 LSTM layers followed by a fully-connected layer and then the output. The model structure can be in Figure 3.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, None, 64)          19968
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                33024
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 57,802
Trainable params: 57,802
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: LSTM Structure

4 Experimental results

4.1 MFCCs features

The results of the neural network training can be seen in Figure 4. Even though overfitting was taken into account with dropout and regularization, it is still present as the accuracy for training keeps increasing steeply compared to the accuracy for testing. The final accuracy is equal to 58%.

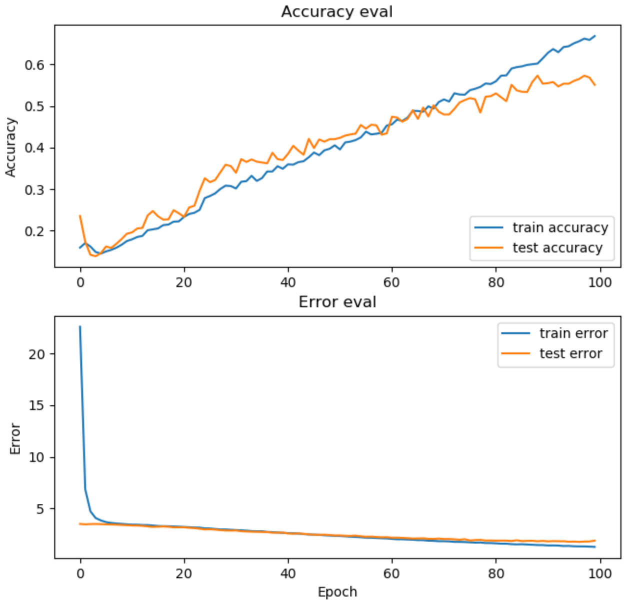


Figure 4: Neural Network Accuracy and Loss

The results of the convolution neural network training can be seen in Figure 5. Less epochs were needed than in the neural network to reach a plateau; the model was trained during 30 epochs and other experiments showed that increasing the number of epochs did not give much better results. The final test accuracy is equal to 71%.

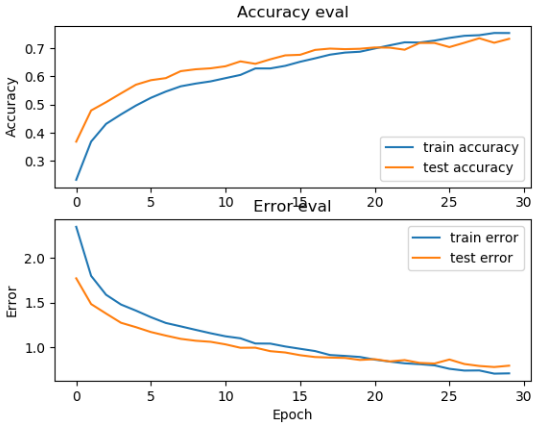


Figure 5: CNN Accuracy and Loss

The accuracy and loss of the RNN-LSTM network training can be seen in Figure 6. It is not as good as the CNN but still an adequate result considering there are 10 different genres (the random accuracy is equal to 10%). Like the first model overfitting can be observed here. The final test accuracy is equal to 61%.

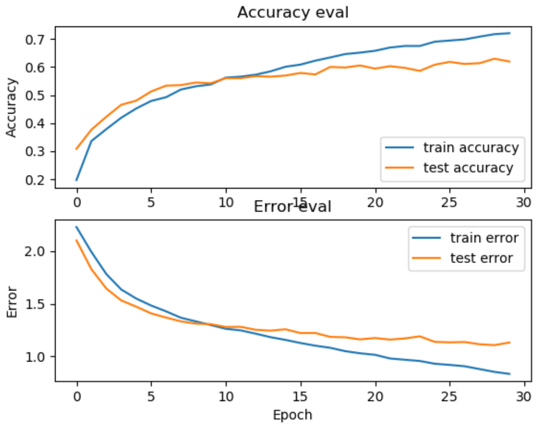
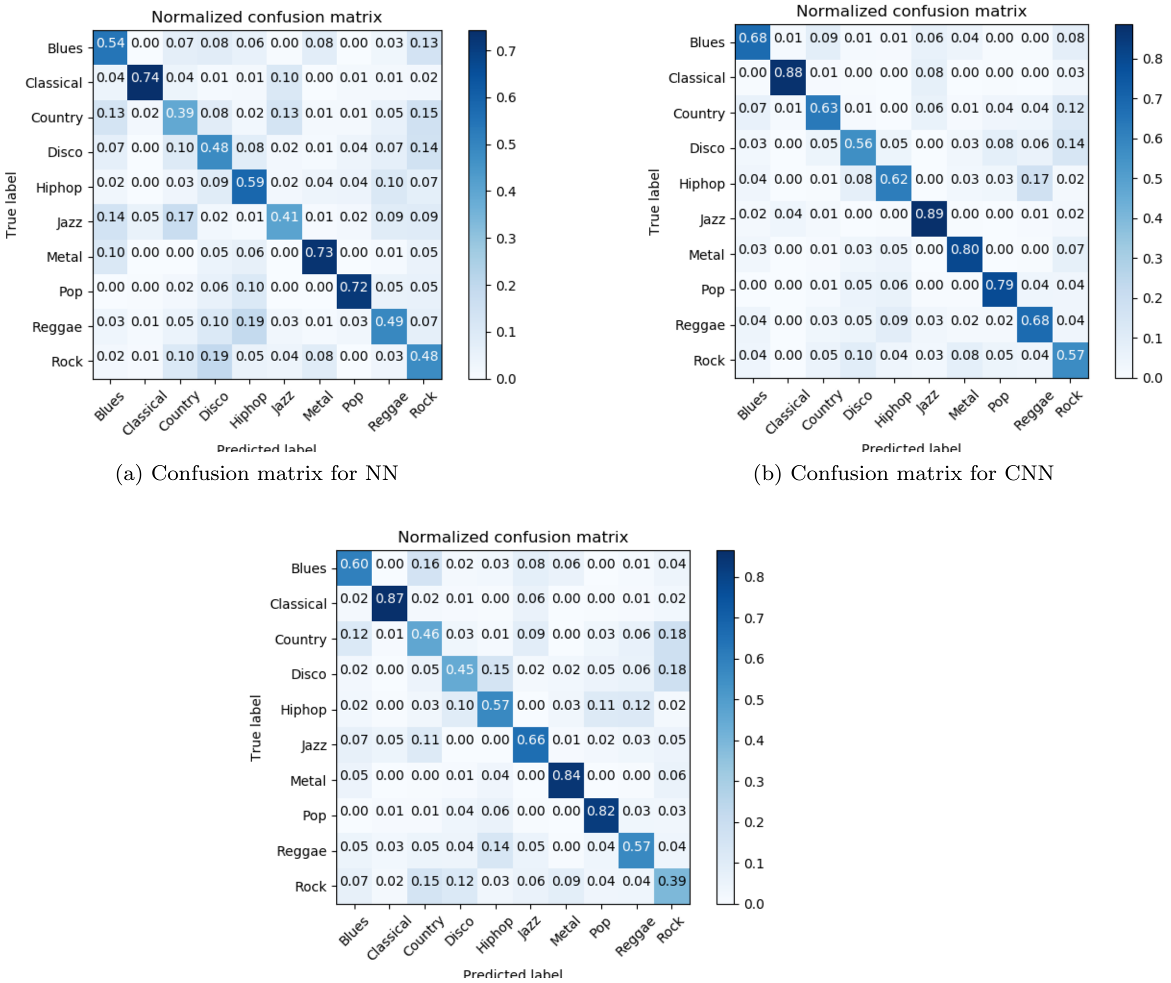


Figure 6: RNN-LSTM Accuracy and Loss

Confusion matrices were computed for all the networks (Figure 7). First of all, it shows like before, that the best accuracy is given by the CNN structure. Also it should be noted that from all three, the same genres seem to be the strongest: Classical, Metal and Pop (except for CNN where Jazz is also a strong genre). Classical makes sense as it is very distinct compared to the others, more structured to the ear. Metal also could be considered very different, very loud. From this we can draw the conclusion that MFCCs are very efficient for music classification and especially for classifying classical, metal and pop music.



(a) Confusion matrix for NN

(b) Confusion matrix for CNN

(c) Confusion matrix for RNN-LSTM

Figure 7: Confusion matrices

4.2 Tempo features

The results of the NN training using tempo as input can be seen in Figure 8. It is clearing not working and it confirms that frequency domain features are much more effective to represent and classify audio.

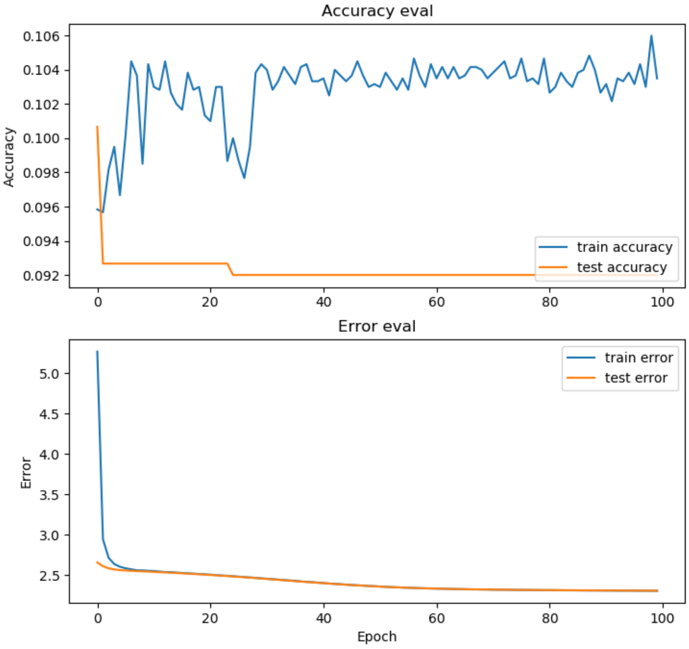


Figure 8: NN Accuracy and Loss for tempo

5 Conclusion

Music can be classified using machine learning techniques. Different features used for inputs and models have been studied for this project to classify genres. CNNs have shown the best results for accuracy and are less prone to overfitting. As for the features, according to research, MFCCs have been the most effective way to represent audio and the experiments have proven it as well.

Of course, there is still room for improvement. Whether it is concerning the features (there are many more and it is possible to combine several of them as well), the models (it is possible to use a hierarchical structure) or even how to classify (genres are known to be subjective and fluctuating).

References

[1] Dieleman, S. and Schrauwen B. 2014. *End-to-end learning for music audio.* IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, 2014, pp. 6964-6968, doi: 10.1109/ICASSP.2014.6854950.

[2] Dong, Mingwen. 2018. *Convolutional Neural Network Achieves Human-Level Accuracy in Music Genre Classification.* 2018 Conference on Cognitive Computational Neuroscience (2018): n. pag. Crossref. Web.

[3] Goulart, A.J.H., Guido, R.C., Maciel, C.D. 2012. *Exploring different approaches for music genre classification.* Egypt. Inform. J. 13(2), 59–63.

[4] Li, Tao, Ogihara, Mitsunori. 2006. *Toward intelligent music information retrieval.* Multimedia, IEEE Transactions on. 8. 564 - 574. 10.1109/TMM.2006.870730.

[5] Oord, A., Dieleman, S., Schrauwen, B. 2013. *Deep content-based music recommendation.* Advances in Neural Information Processing Systems.

[6] Slot, Krzysztof. *Speech Recognition: Speech signal descriptors.* Course slides. Instytut Informatyki Stosowanej, Politechnika Łódzka.

[7] Tang, Chun Pui, Ka Long Chui, Ying Kin Yu, Zhiliang Zeng, and Kin Hong Wong. 2018. *Music Genre Classification Using a Hierarchical Long Short Term Memory (LSTM) Model.* In Third International Workshop on Pattern Recognition, 10828:108281B. International Society for Optics and Photonics. https://doi.org/10.1117/12.2501763.

[8] Tzanetakis G, Cook P. 2002. *Musical genre classification of audio signals.* IEEE Trans Speech Audio Process 2002;10(5):293–302.

[9] Velardo, V. 2020. *Deep Learning (for Audio) with Python.* URL = https://www.youtube.com/playlist?list=PL-wATfeyAMNrtbkCNsLcpoAyBBRJZVlnf