# Music Genre Classification: CNN and CRNN

Christina Gaitanou
*Advanced Machine Learning*
*IT University of Copenhagen*
Copenhagen, Denmark
cgai@itu.dk

Faye Sabine Hahn
*Advanced Machine Learning*
*IT University of Copenhagen*
Copenhagen, Denmark
fhah@itu.dk

Honorata Marta Pechal
*Advanced Machine Learning*
*IT University of Copenhagen*
Copenhagen, Denmark
hope@itu.dk

Maja Ørslund
*Advanced Machine Learning*
*IT University of Copenhagen*
Copenhagen, Denmark
oers@itu.dk

## I. Introduction

Music Genre Classification is an important task and broadly used in many products and services daily. All music providers provide music classification or recommendation to their customers [Ramgude, 2021]. Audio classification has been a more difficult task for a software program than from a human perceptive since audio signals have to been transformed into a representation that can be later processed by a software associated with the relevant meaning or label. Today, Deep Learning methods have been proven the most effective and efficient for these music genre classification problems [Ramgude, 2021]. For the purpose of this project we are transforming and preprocessing audio signals into visual representations so that they can be used in training a Neural Network aiming to extract feature maps that will later be used to classify new data.

Recurrent Neural Networks (RNNs) that have been used in sequential data, are combined with Convolutional Neural Networks (CNNs) to form a hybrid model as Convolutional Recurrent Neural Network (CRNN). CNNs are often used to solve music genre classification tasks, and more recently so have CRNNs [Choi et al., 2016]. This paper investigates and compares the performance of a Convolutional Neural Network (CNN) and Convolutional Recurrent Neural Network (CRNN) for the Music Genre Classification problem. For building these models we are experimenting with the baseline as found in research, to achieve the best possible performance. Furthermore, for this comparison, we are using the GTZAN dataset that consists of 100 audio files for each of the 10 classes. On this dataset, we are experimenting with different appropriate methods for preprocessing that are later used with the chosen models (CNN and CRNN) and comparing the results.

The main goal of this project is to use state-of-art deep learning methods to extract features from audio signals in order to classify the data into labels of the relevant music genres. More specifically, it is focused on the comparison and performance of two models CNN and CRNN, with 10 different classes using different versions of the GTZAN dataset.

## II. Methods

This section describes in more detail, the deep learning models' architecture used to solve a single label genre classification problem. Many forms of research show that a CNN model outperforms other machine learning feature extraction methods [Bahuleyan, 2018]. This framework is very effective since it is able to learn hierarchical representations of high-dimensional data. More recently, a CRNN has proved to be even more efficient than a CNN [Choi et al., 2016]. The CRNN and CNN have the exact same architecture and structure except a distinctive difference with the addition of recurrent layers in the CRNN. During supervised learning, CNNs extract features by convolutional kernels while the additional recurrent layers of CRNN play the role of a temporal summariser [Choi et al., 2016]. The comparison is based on the consideration of accuracy as the main performance metric.

### A. CNN

Convolutional Neural Network is a neural network designed for processing data that have a known grid-like topology [Goodfellow et al., 2016]. The convolution process applies a set of filters to these images (2D pixels), in order to extract features at different scales, and outputs a set of feature maps. The output of every 2D convolutional layer is followed by an activation function to make the network learn complex patterns and 2D max pooling to further reduce the dimensionality of the features. In this experimentation the ReLU (Rectified Linear Unit) activation function showed the best results. After flattening the output, two fully connected layers follow. The first dense layer also uses ReLU activation function to decide which features are relevant for which classes. The last fully connected layer, composed of 10 units, is using the softmax activation function to assign the final output as a probability distribution over the possible classes. The input is thus classified as the genre with highest probability. Additionally, after every convolution layer batch normalization is implemented for faster convergence and a more stable model. After some experiments, we finalised on dropout layers of 10 % after each convolutional layer, and 30% after the dense layer to improve the model's accuracy. For the implementation of the layers for this framework we are using the Keras library [Chollet et al., 2015]. The final holistic architecture for the CNN is shown in figure 1 which includes number and size of filters.

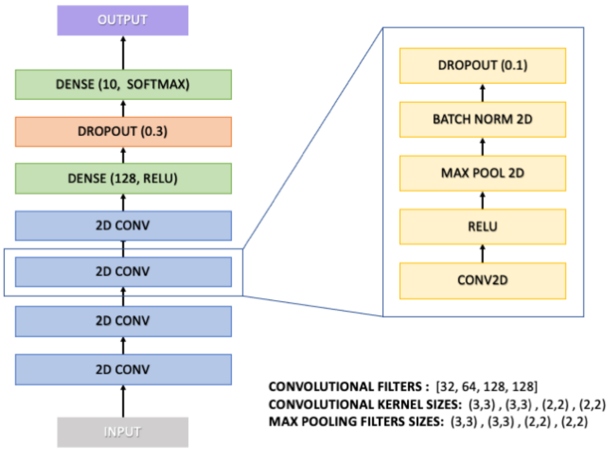

OUTPUT
DENSE (10, SOFTMAX)
DROPOUT (0.3)
DENSE (128, RELU)
2D CONV
2D CONV
2D CONV
2D CONV
INPUT
DROPOUT (0.1)
BATCH NORM 2D
MAX POOL 2D
RELU
CONV2D
CONVOLUTIONAL FILTERS : [32, 64, 128, 128]
CONVOLUTIONAL KERNEL SIZES: (3,3) , (3,3) , (2,2) , (2,2)
MAX POOLING FILTERS SIZES: (3,3) , (3,3) , (2,2) , (2,2)

Fig. 1: CNN architecture diagram

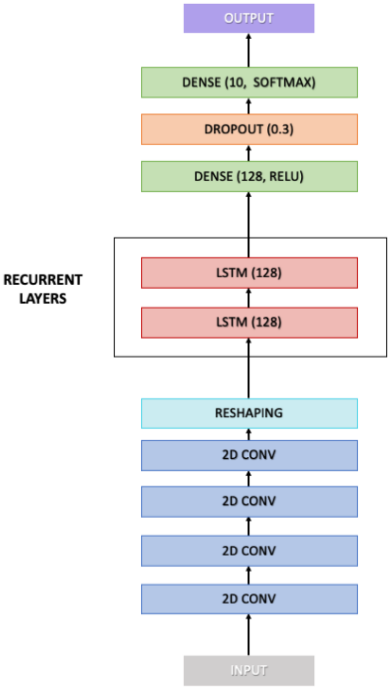


Fig. 2: CRNN architecture diagram

### B. CRNN

For the CRNN, we continue with the CNN model as in figure 1. The two recurrent layers is added between the convolutional layers and the fully connected layers. Whereas the convolutional layers extract features from the visual representations of the audio signals, the recurrent layer is used to correlate the captured temporal dependencies between the output data of the last convolutional layer. For this music genre classification problem the CRNN framework has the distinctive ability to capture both the spectral features and temporal patterns in the visual representation of audio signals [Choi et al., 2016].

The recurrent layers we experiment with are the Long Short-Term Memory (LSTM) and Gated Recurrent (GRU) networks. These models are designed and broadly used for processing long sequential data by using gates to control the flow of information through the network over a long period of time. The main difference between LSTMs and GRUs is the application of these gates. LSTMs control the flow of information using three different gates (input, output and forget) whereas GRUs use two gates (reset and update). This CRNN adaptation is using two LSTM recurrent layers instead of GRU layers since our experimentation showed that this framework provided slightly better performance. The final architecture for the CRNN model is shown in figure 2. This implementation took a lot of trials and errors and reshaping the output of the convolutional layer to the proper input for the recurrent was crucial. The reshaping process is important to change for every different kind of input data used in the network accordingly for the recurrent layers to work.

## III. Experiments

### A. Dataset

We use the GTZAN dataset [Tzanetakis et al, 2001], which contains a 1000 labeled 30 second tracks as mp3 files. The 30 seconds are extracted from the highlight of the track. The 1000 tracks are divided into 10 genres, with 100 tracks for each genre. The genres are: pop, metal, disco, blues, reggae, classical, rock, hiphop, country and jazz. To get an overview of the genre similarities, we present a principal components analysis in figure 3. From this we note that some genres e.g. classical and metal are more clustered and concentrated on each their principal component, but genres such as rock varies across both principal components.

The dataset is split into a train-, validation- and testset in the ratio 80:10:20. As we wish to test models on the same data structure as we are training on, it is important that all ten genres are evenly distributed within all three data splits. As the data is structured by first listing all 100 pop tracks, then all metal tracks, etc. the data is shuffled before the split. As can be seen in figure 4, the genres are observed to be quite balanced in the three datasets, with a slight variability in country and jazz.

### B. Features

To train the model we will not use the raw waveform representation of the data. As we operate with CNNs for

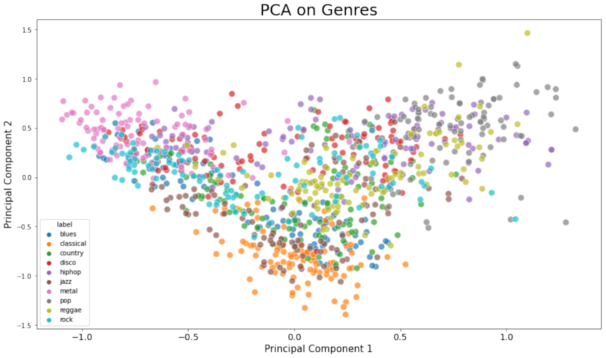


Fig. 3: PCA of the GTZAN dataset

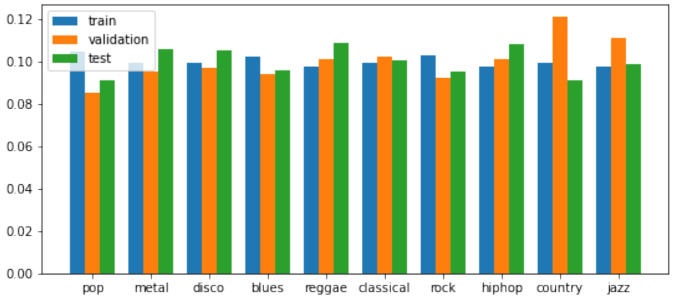


Fig. 4: Distribution of genres in training, validation and test set

feature extraction of both models, it is necessary to use an image representation of the data. State-of-the-art for genre classification is to either use mel-spectrograms or mel-frequency cepstrum coffiecients (denoted ”MFCCs” in the rest of the report), which are derived from the mel-spectrograms. Both formats are designed to mimic the way humans perceive sound.

Audio is represented as waves in a time-amplitude domain. We can convert to a time series by sampling the amplitude at regular time intervals. This is called the sampling rate, which is defined as number of samples per second (Hz). All tracks in the GTZAN dataset have been sampled using a sampling rate of 22.050 Hz. The sampling rate controls size of input, so if a dataset consists of tracks measured with inconsistent sampling rates, pre-processing needs to be done. We want to extract the frequency, at each time step. The frequency is the pitch of a sound, i.e. how deep or high the sound is perceived. From the time-amplitude domain the wave is converted to a spectrogram which has exactly the time-frequency domain we are looking for. This is done using Fourier transformation. However, humans unfortunately cannot distinguish frequencies similarly across the entire spectrum, and furthermore we are not physically able to perceive all frequencies. Thus, we convert the spectrograms to mel-spectrograms, that has the time-melscale domain and also uses decibel instead of amplitude, which is more intuitive for humans. Finally, from the mel-spectrograms the MFCCs are derived by taking the discrete cosine transform [Doshi, 2021]. As we have limited training data, we want to reduce number of dimensions of features to prevent overfitting. Hence, we will use MFCCs to train the model, as it performs a smoothing operation on mel-spectrograms and consequently reducing dimension size of input. An example of an MFCC representation of a track can be seen in figure 5. On the first axis is the number of time bins the track is separated into, and on the second axis is the number of coefficients. The coefficients are similar to principal components in the way that the first coefficients capture most of the relevant information. As seen in figure 5, the coefficient value is essentially an indication of how well each coefficient fits the mel-spectrogram [Velardo, 2021].

An intuitive way to understand how the MFCC representation works in a CNN, is to think of it as a long rectangular image, where the height (coefficients) and width (time bins) represent the number of pixels and the value of each coefficient for each time step as the pixel value. The height and width of the image depend on the chosen number of coefficients and the length of the track. The image only has one channel, like a grey-scale image, as the audio is mono.

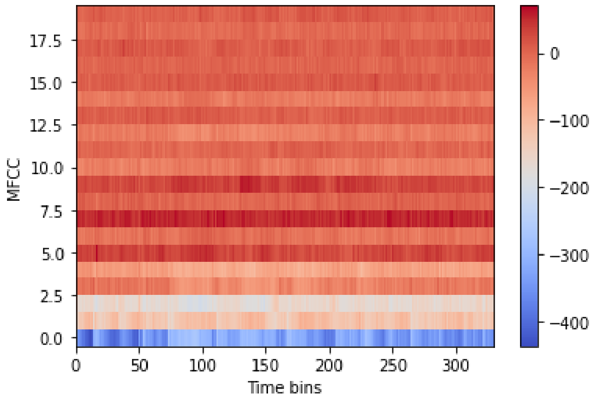


Fig. 5: MFCC representation of a 3 second extract of a blues song with 331 time bins on first axis and 20 coefficients on second axis, thus making it a (331x20) ”pixel” image

### C. *Finetuning training data*

CNNs and CRNNs will with a few layers quickly develop a considerable amount of trainable parameters. One potential issue with having a larger number of hyperparameters is overfitting. If there is not enough training data to properly finetune parameters, the model will overfit and as a consequence lack in generalizability. The number of hyperparameters can be regulated either by controlling the size of the input data (e.g. through the number of coefficients or number of time bins in the MFCC representation), or by the architecture of the models. As the original training dataset only contains 699 samples, we aim to optimize the models' performance and mitigate overfitting by experimenting with different versions of the training dataset. The variation approach for the data includes controlling for the size of each data point and adjusting the number of instances in the training set. The overall objective of this is to experiment with the size and complexity of the data to achieve a higher generalizability of the model. Therefore, we ultimately varied in the number of MFCC coefficients, track lengths and in the use of augmented data versus non-augmented data. An overview of the different datasets variants, their effect on the number of hyperparameters and the resulting accuracies for each model can be seen in table I.

It is to note that the data augmentation is performed only after the splitting the data, as it is only executed on the training data. We augment the data in four different ways, such that the size of the training data is quadrupled. The augmentation is done using an audio augmentation library from Spijkervet [Spijkervet, 2021].

The models' hyperparameters are trained using an Adam optimizer, which uses a stochastic gradient to update the weights [Chollet et al., 2015]. The optimizer is used with a learning rate of 0.001 and a sparse categorical cross-entropy loss function. The advantage of this loss function over the vanilla categorial cross-entropy is that the computational cost is lower since the target classifications are utilized as integers instead of using an one-hot encoding approach. The loss of the hyperparameters (model weights) is calculated as:

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)),$$

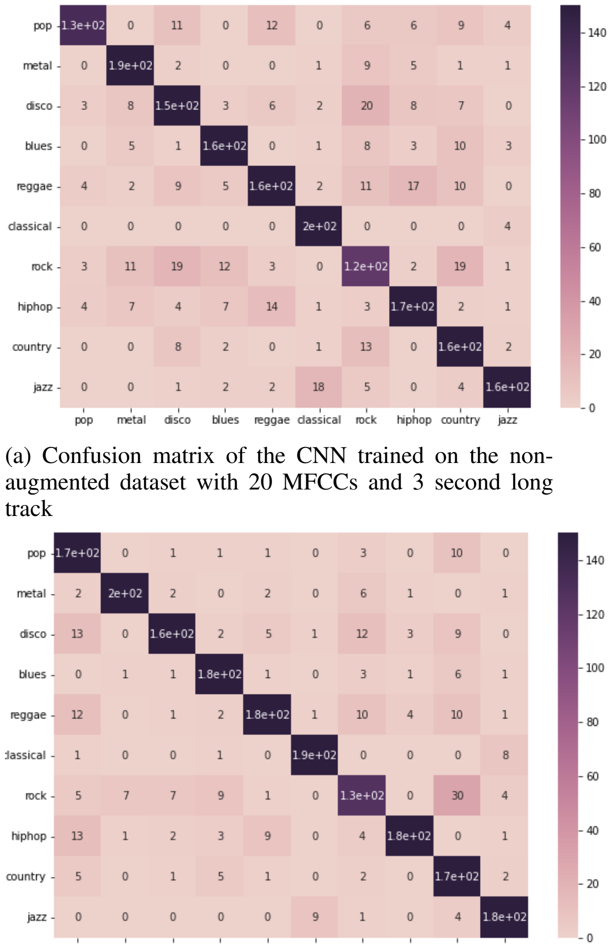
where $y_i$ represent the true label and $\hat{y}_i$ is the predicted label.

We use a batch size of 32 and the metric we evaluate the models on is the accuracy. We choose to only consider accuracy as we argue this metric is not biased for this problem since the dataset is well balanced in terms of genre labels. We set the number of epochs to 30.

## IV. Discussion

### A. Results

Table I shows the results for CNN versus CRNN for each dataset variant. Each row contains the performance indicators for the corresponding CNN and CRNN architecture, where the only difference is two additional LSTM layers in the CRNN. Both models perform best when trained on the dataset variant with an input size of (331, 20), i.e. the smallest input size of 3-second long tracks and 20 MFCC coefficients. The CNN model performs better with non-augmented data, with a test accuracy of 80% in the best case. The results indicate overfitting, as the train accuracy is significantly higher at 96%. The equivalent CRNN model shows slightly less overfitting and obtained a higher test accuracy. On the other hand, the better input variant in the before-mentioned configuration is augmented data for the CRNN. Here, the model performs better on the test set (87% accuracy) than on the train set (80% accuracy). Yet again, the analogous CNN is overfitting. Liu et. al (2019) implemented a novel CNN architecture, called BBNN, on GTZAN along with two other datasets. The highest classification accuracy was 93.9% and was obtained on Mel-spectrograms [Liu et al., 2019]. Another study run a two phase hybrid classifier on GTZAN and achieved classification accuracies of 88.3%. The input data was MFCC and the focus of the study was to solve the problem of misclassification of certain genres [Karunakaran and Arya, 2018]. This shows that our model is competitive with models existing in the literature.



(a) Confusion matrix of the CNN trained on the non-augmented dataset with 20 MFCCs and 3 second long track

(b) Confusion matrix of the CRNN trained on the augmented dataset with 20 MFCCs and 3 second long track

Fig. 6: Confusion matrix of both models with highest accuracy

### B. Confusion matrix

Figure 6 shows the confusion matrices for the best-performing CNN and CRNN models. Generally, CRNN is more accurate at classifying music genres. However, the mistakes made by both models are very similar. For example, both models tend to misclassify jazz with classical, but the CNN model does so twice as often. Both models are best at classifying classical and metal. Rock was the genre that seems to be most difficult to classify for both models. CNN mislabeled rock as many different genres and CRNN repeatedly confused rock with country. Moreover, both models face difficulties with the classification of pop and disco. CRNN confuses disco with pop and rock. CNN repeatedly misclassifies disco as rock and confuses jazz with classical. By looking at figure 3, some explanation can be found. Jazz and classical are placed very close to each other, which possibly means that their frequency information is alike. Rock seems to be extremely scattered and overlaps with many different genres.

### *C. Strengths and weaknesses of the models*

Both models are strong in extracting enough features and patterns from rather short (3-second long) audio sequences to classify the music genre with fairly high accuracy. The advantage of the CNN model for our application is that it is less computationally expensive than the CRNN while achieving a still decent performance. The CRNN on the other hand stands out with its combined ability to first extract the local features (in the convolutional layers) and later capture long-term dependencies by processing the data in a sequential manner (in the recurrent layers) [Murphy, 2022]. Combining these strengths, the CRNN is able to outperform the CNN.

The main drawback of the CNN model is its proneness to overfitting. This even resulted in lower accuracy for the augmented dataset variants (as can be seen in table I) as the CNN tended to overfit the noisy features added during augmentation. Some closer inspection of this phenomena showed that the only way to improve validation and test accuracy on those variants was to force down train accuracy by e.g. increasing drop-out or reducing the number of learned features.

But even when leaving out this factor, some level of overfitting could be observed for almost all model variations. We tried different approaches to overcome this obstacle: adding dropout layers at every layer and experimenting with different dropout rates, decreasing the number of trainable parameters by reducing the number of layers, decreasing the number of filters, and increasing stride (step size).

### *D. Shortcomings of the dataset*

There are multiple limitations to the GTZAN dataset which were already explored in [Tzanetakis et al, 2001]. Firstly, Tzanetakis et al. found that GTZAN does not reflect the real-life complexity of music genres. There are many more genres than the 10 genres represented in GTZAN. Secondly, they mentioned that the existence of subgenres and overlapping genres is ignored. The genres in the dataset are clearly divided and no data points that lay on the intersection of different genres are represented. Thirdly, one can argue that the size of 1000 tracks is not significant enough to produce significant outcomes. Finally, each track is only 30 seconds long. Thus, Tzanetakis et al. argue that the full complexity of each track is not depicted. Despite the aforementioned weaknesses, the dataset is still widely used in the literature and has repeatedly served as a means of comparing different models in academia. The results obtained on that dataset are absolutely relevant. Nevertheless, the limitations should be kept in mind.

### *E. MFCCs*

MFCCs are a manually-designed depiction of spectral information. This representation of audio data was designed by people with extensive knowledge about how humans perceive sound [Cohen and Gannot, 2008]. While it may be intuitive to mimic human perception in the first place, it might also introduce some bias. An alternative approach could therefore be to let the model decide which features are most relevant by providing a (more) raw representation of the audio file. This was already found to be a viable solution in the case of a large-scale classification of bird sounds where the model performed better on raw audio rather than on MFCCs [Stowell and Plumbley, 2014].

### *F. Data augmentation*

Telling from the results depicted in table I, we can see that data augmentation do not always lead to improved performance of the model. Especially in the case of the CNN it often even decreases the test accuracy to augment the data prior to

TABLE I: Overview of different training datasets and their affect on accuracy and hyper parameters

| **Train set** | **Size of train set** | **Input size (time bins, coefficients, channel)** | **Model** | **Total number of trainable parameters** | **Train accuracy** | **Test accuracy** |
|---|---|---|---|---|---|---|
| Full track, 40 coefficients, augmented data | 2793 | (3308,40,1) | CNN—CRNN | 6.869.706 — 399.050 | 0.33 — 0.73 | 0.26 — 0.08 |
| Full track, 40 coefficients, non-augmented data | 699 | (3308,40,1) | CNN—CRNN | 6.869.706 — 399.050 | 0.96 — 0.90 | 0.6 — 0.4 |
| Full track, 20 coefficients, augmented data | 2793 | (3308,20,1) | CNN—CRNN | 3.494.602 — 399.050 | 0.98 — 0.73 | 0.35 — 0.58 |
| Full track, 20 coefficients, non-augmented data | 699 | (3308,20,1) | CNN—CRNN | 3.494.602 — 399.050 | 0.99 — 0.85 | 0.5 — 0.58 |
| 3 second track, 40 coefficients, augmented data | 27972 | (331,40,1) | CNN—CRNN | 774.858 — 399.050 | 0.91 — 0.82 | 0.77 — 0.82 |
| 3 second track, 40 coefficients, non-augmented data | 6993 | (331,40,1) | CNN—CRNN | 774.858 — 399.050 | 0.96 — 0.93 | 0.74 — 0.72 |
| 3 second track, 20 coefficients, augmented data | 27972 | (331,20,1) | CNN—CRNN | 447.178 — 399.050 | 0.90 — 0.80 | **0.77 — 0.87** |
| 3 second track, 20 coefficients, non-augmented data | 6993 | (331,20,1) | CNN—CRNN | 447.178 — 399.050 | 0.96 — 0.93 | **0.80 — 0.83** |

training which is opposite to what we anticipated beforehand. One possible reason for that could be that the augmentation technique introduces too much noise and variability. This, in turn, might contribute to information loss as well as confusing the model to a certain degree. Moreover, a study conducted by Zhao and Wang shows that MFCCs are not robust to additive noise [Zhao and Wang, 2013]. Thus, there is a chance that applying data augmentation could be more fruitful if another audio data representation is chosen.

### *G. Future work*

Further research with our work as the starting point can go in multiple directions. New variations of the network architecture along with optimization techniques can be explored. For example, implementing deeper networks, using different types of layers, and adjusting the model with the use of different optimization techniques (e.g. drop-out, regularization). Another approach could delve into incorporating additional datasets into the model, as GTZAN has a lot of shortcomings. Finetuning our model on another dataset could lead to better performance and improved generalizability while at the same time-saving time during training. Thirdly, more raw audio representations can be tested. Lastly, the interpretability of the results can be investigated e.g. by decoding which features are decisive for classifying a song as a certain genre.

## V. Conclusion

In this project we build CNN and CRNN deep learning models to solve a music genre classification problem using the GTZAN dataset, aiming to compare and improve their performance. To achieve this, we experimented with different model architectures and data preprocessing techniques. The overall results show that the dataset consisting of a short 3 seconds track in 20 MFCC coefficients obtains the best performance in terms of train and test accuracy and CRNN outperforms CNN. In particular, the accuracy of the CRNN was higher for the augmented data. On the contrary, the nonaugmented data worked better on CNN. Future possible experimentation could be to use more diverse and bigger datasets and build even deeper networks.

## References

[Bahuleyan, 2018] Bahuleyan, H. (2018). Music genre classification using machine learning technique.

[Choi et al., 2016] Choi, K., Fazekas, G., Sandler, M., and Cho, K. (2016). Convolutional recurrent neural networks for music classification.

[Chollet et al., 2015] Chollet, F. et al. (2015). Keras.

[Cohen and Gannot, 2008] Cohen, I. and Gannot, S. (2008). *Springer Handbook of Speech Processing*.

[Doshi, 2021] Doshi, K. (2021).

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Karunakaran and Arya, 2018] Karunakaran, N. and Arya, A. (2018). A scalable hybrid classifier for music genre classification using machine learning concepts and spark. *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, pages 128–135.

[Liu et al., 2019] Liu, C., Feng, L., Liu, G., Wang, H., and Liu, S. (2019). Bottom-up broadcast neural network for music genre classification.

[Murphy, 2022] Murphy, K. P. (2022). *Probabilistic Machine Learning*. The MIT Press, Cambridge MA, 1 edition.

[Ramgude, 2021] Ramgude, A. (2021). Music genre classification.

[Spijkervet, 2021] Spijkervet, J. (2021). Spijkervet/torchaudio-augmentations.

[Stowell and Plumbley, 2014] Stowell, D. and Plumbley, M. D. (2014). Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *CoRR*, abs/1405.6524.

[Tzanetakis et al, 2001] Tzanetakis et al (2001). Automatic Musical Genre Classification Of Audio Signals. https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification.

[Velardo, 2021] Velardo, V. (2021).

[Zhao and Wang, 2013] Zhao, X. and Wang, D. (2013). Analyzing noise robustness of mfcc and gfcc features in speaker identification. pages 7204–7208.