Music Genre Classification using Machine Learning

TDDE19

Group 7
Lisa Spahn Lundgren (lissp373)
Emil Bråkenhielm (emibr678)
Olivia Shamon (olish585)
David Ångström (davan288)
Drinas Kastrati (drika827)
Daniel Hu (danhu028)

Contents

1 Introduction

People like to listen to different genres of music such as country, rock, or pop. In today's digital world, applications such as Spotify, Google Music, and Apple Music allow people to connect and listen to their favorite songs with the click of a button. Thousands upon thousands of hours of new music content is uploaded to these music platforms each day, music content that may not always have a genre label, or which may in some cases be mislabeled. It would be very labor-intensive and time-consuming for someone to go through, listen to, and label each song. Being able to correctly label these songs automatically would save a lot of manpower and resources.

1.1 Background

In machine learning and statistics, classification is a type of supervised learning in which a model learns from a given set of training data and is able to make predictions on data it has never seen before based on what it has previously learned. Problems can be binary classification problems or multi-class problems such as in the case of this project. Classifying problems can also relate to many different areas such as text, image, or even speech recognition. There are many different classification algorithms that can be used, each with its advantages and disadvantages. The performance of these algorithms varies depending on the classification problem, some algorithms perform better for certain types of problems. The tuning of the hyper-parameters and the quality of the data set used will also affect the results. There may also be trade-offs to be made between training time and the achieved accuracy.

There are thus many approaches for music genre classification. One of these approaches is converting audio files into suitable representation of features and training a model to recognize these features. The k-Nearest Neighbours (KNN) algorithm and the Support Vec- tor Machine (SVM) algorithm have been shown to perform well in music genre classification on the Spotify music data set with SVM having an accuracy of 80% and KNN having an accuracy of 77.18% [7]. Another approach is using a Neural Network, more specifically a Convolutional Neural Network (CNN). CNNs are designed for image classification problems and one approach is to train a model on a visual representation of audio files by converting the audio files to spectrograms.

1.2 Aim

The aim of this work is to explore different machine learning approaches to music genre classification. Furthermore, to explore which of the machine learning techniques selected (CNN, KNN, and SVM) is best suited in regards to the accuracy of predictions. It is also of interest to analyze the genres to see which are easiest to classify as well as which are harder

to classify and why. Furthermore, it is interesting to see which features in the training data have the highest impact on the classification of genres.

1.3 Delimitations

As there are countless sub-genres, a decision was made to classify only the most common genres. The ten following genres were selected: Blues, Classical, Country, Disco, Hip hop, Jazz, Metal, Rock, Reggae, and Pop.

There are also many different CNN architectures, which gives a delimitation of the results and conclusion, which can not be generalized to all CNNs, but only to the specific CNN architecture used in this study.

2 Dataset

GTZAN [12] is a public dataset containing songs from 9 different genres, 100 songs for every genre. Each song is mapped to its corresponding genre. Additionally, spectrograms for the 30s audio files already exist, as well as features for both 30s and 3s audio samples. Figure 1 shows an example of all features for one audio file. Figure 2 illustrates a sample spectrogram of an audio file in the blues genre. The values on the x -and y-axis are not labeled or scaled as they are irrelevant for the image-based classification.

GTZAN - 3 sec features

filename	**length**	**chroma_stft_mean**	**chroma_stft_var**	**rms_mean**	**rms_var**
blues.00000.0.wav	66149	0.3354063630104065	0.09104829281568527	0.1304050236940384	0.0035210042260587215
spectral_centroid_mean	**spectral_centroid_var**	**spectral_bandwidth_mean**	**spectral_bandwidth_var**	**rolloff_mean**	**rolloff_var**
1773.0650319904662	167541.6308686573	1972.7443881356735	117335.77156332089	3714.560359074519	1080789.8855805045
zero_crossing_rate_mean	**zero_crossing_rate_var**	**harmony_mean**	**harmony_var**	**perceptr_mean**	**perceptr_var**
0.08185096153846154	0.0005576872402394312	-7.848480163374916e-05	0.008353590033948421	-6.816183304181322e-05	0.005535192787647247
tempo	**mfcc1_mean**	**mfcc1_var**	**mfcc2_mean**	**mfcc2_var**	**mfcc3_mean**
129.19921875	-118.62791442871094	2440.28662109375	125.08362579345703	260.9569091796875	-23.443723678588867
mfcc3_var	**mfcc4_mean**	**mfcc4_var**	**mfcc5_mean**	**mfcc5_var**	**mfcc6_mean**
364.08172607421875	41.32148361206055	181.69485473632812	-5.976108074188232	152.963134765625	20.115140914916992
mfcc6_var	**mfcc7_mean**	**mfcc7_var**	**mfcc8_mean**	**mfcc8_var**	**mfcc9_mean**
75.65229797363281	-16.04541015625	40.22710418701172	17.85519790649414	84.32028198242188	-14.633434295654297
mfcc9_var	**mfcc10_mean**	**mfcc10_var**	**mfcc11_mean**	**mfcc11_var**	**mfcc12_mean**
83.4372329711914	10.270526885986328	97.00133514404297	-9.70827865600586	66.66989135742188	10.18387508392334
mfcc12_var	**mfcc13_mean**	**mfcc13_var**	**mfcc14_mean**	**mfcc14_var**	**mfcc15_mean**
45.10361099243164	-4.681614398956299	34.169498443603516	8.417439460754395	48.26944351196289	-7.233476638793945
mfcc15_var	**mfcc16_mean**	**mfcc16_var**	**mfcc17_mean**	**mfcc17_var**	**mfcc18_mean**
42.77094650268555	-2.8536033630371094	39.6871452331543	-3.2412803173065186	36.488243103027344	0.7222089767456055
mfcc18_var	**mfcc19_mean**	**mfcc19_var**	**mfcc20_mean**	**mfcc20_var**	**label**
38.099151611328125	-5.05033540725708	33.618072509765625	-0.24302679300308228	43.771766662597656	blues

Figure 1: Features in the GTZAN dataset for a blues sample file.

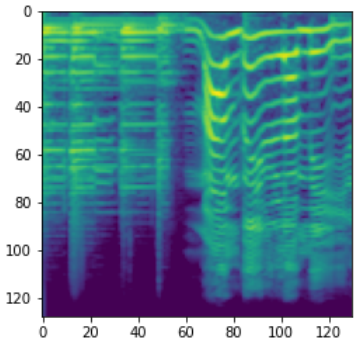


Figure 2: Example of a blues spectrogram.

3 Method 1 - Feature-based Classification

One approach to predict the genre of a song is to do feature-based classification. For this, different features of a song are extracted and this numerical feature representation can then be used to classify the song.

3.1 Features

Each song in the GTZAN data set has 57 extracted features from the raw audio file. They may be divided into those related to time, which are extracted directly from the audio signal, and those related to the frequency which comes from the audio files' cepstral representation. Examples of features are Tempo, which is measured in beats per minute (BPM) the songs speed, and Zero Crossing Rate, which is related to the discrete-time signal and the frequency at which it changes sign [9]. For each feature, the mean and the variance over the entire audio file are calculated.

3.2 Preprocessing

In the raw data, the features had very different scales, ranging from magnitudes around 10^{-6} to almost 10^{7}. This can be a problem when using distance-based classification methods such as k-Nearest-Neighbors or Support Vector Machines since features that have larger magnitudes will influence the distance a lot more than lower-magnitude features [8]. But it does not necessarily have to be the case that these variables have a larger effect on the genre than lower-scale variables. In order the make the features comparable, we decided to standardize the data. This means we scale each covariate to have a mean of zero and a standard deviation of one. The new data can be calculated as follows

$$z = \frac{x - \mu_f}{\sigma_f}, \tag{1}$$

where x is one observation of feature f, μ_f is the sample mean of feature f and σ_f is the standard deviation of the feature [8]. The mean and standard deviation are calculated from the raw data. z is then the standardized observation.

3.3 K-Nearest Neighbor

The k-nearest neighbor classifier (KNN) is a commonly used classification method, that is popular due to its efficiency despite being a rather simple approach [5]. Given a number of n data samples $x_1, x_2, ...x_n$, the class of a new input sample x' is predicted by calculating the pair-wise distance between x' and each data sample x_i. Then, the k closest neighbors are identified and the class label of x' can be predicted by the majority rule, meaning which is the most common class amongst the k nearest neighbors.

The choice of how many neighbors k are considered strongly influences the behavior and performance of the classifier [10]. In general, smaller k-values yield more complex decision boundaries, but increase the risk of overfitting and can be more sensitive towards outliers. Larger values for k will create smoother decision boundaries but can result in an underfitted model. Hence a common procedure to select the best k-value is to perform cross-validation over multiple folds [10].

Another element that influences the classification performance is the distance measure that is used to calculate the pair-wise distance between two data samples [2]. Which distance measure is chosen depends on the problem and how the similarity between two points should be measured, but the most common metric is the Euclidean distance [10]. For two data points x' and x with m features, the Euclidean distance d between those points is defined as follows

$$d(x', x) = \sqrt{\sum_{i=1}^{m}(x'_i - x_i)^2} = \sqrt{(x'_1 - x_1)^2 + ... + (x'_m - x_m)^2} \ . \tag{2}$$

3.4 Support Vector Machine

Support Vector Machines (SVM) is another machine learning model used to classify inputted data to some previously classified data. It does so by viewing each data point x as a p-dimensional vector, where p is the number of features for x. While training, Hyper-planes (decision boundaries) of the same dimension as the p-dimensional vector are created with the goal of separating the data points based on their classification. The support vectors are then placed in relation to the hyper-plane to decide the position and orientation of the hyper-plane. It is through these vectors that the margin can be maximized between the differently classified data points.

To decide the positions of the data points, a kernel has to be used to map the p-dimensional vectors into a different space. In our case, since the relations between genres may not be linear, the Gaussian kernel was selected.

$$K(x, x') = exp(-\frac{||x - x'||^2}{2\sigma^2}) \tag{3}$$

where x and x' are p-dimensional vectors, d is the dimensions and σ is a marginalization parameter. For large values of σ, the margin between the support vectors and data points will be smaller and conversely, for smaller values of σ, the margin will be larger.

3.5 Feature Analysis

It can be of interest to visualize the data, to get an idea of what the data looks like. However, since our data has 57 dimensions, it is useful to reduce the dimensionality of the data. This

can be achieved using Principal Component Analysis (PCA) [1]. PCA maps the original data to a lower-dimensional coordinate space by creating new variables, so-called principal components while trying to preserve as much information as possible. For instance, if variables are linearly correlated, it is sufficient to keep only one of those variables to describe the general trend of the data.

Mathematically, this can be done by first calculating the covariance matrix of the data, to identify correlated features. Next, the principal components are calculated as combinations of important features. The first principal component is the one that describes the maximum variance of the data, i.e. it holds the most information. The second principal component describes the direction of the second largest variance, and so on. Hence to describe the general behavior of the data, it is often sufficient to look at the first two principal components, meaning that the dataset is mapped to two dimensions, which can easily be visualized.

In order to further analyze the features, it can be interesting to rank the features by importance. For this, Linear Discriminant Analysis (LDA) [4] is used, which is similar to PCA in the sense that it maps the data to a lower-dimensional domain, and then tries to create a linear decision boundary between the classes given the transformed data. LDA transformation maximizes the separability between the different classes, by mapping the data to a domain where the means of the classes are as far apart as possible, while also minimizing the variance of the data points within each class. Hence the features that are the most weighted by LDA can be interpreted as the features that enable the most separability of the data and hence are the most relevant for our problem.

4 Method 2 - Image-based Classification on Spectrogram

For the image-based classification of music genres, the data needs to be preprocessed before being fed into the CNN model. Figure 3 shows each step of the proposed method.

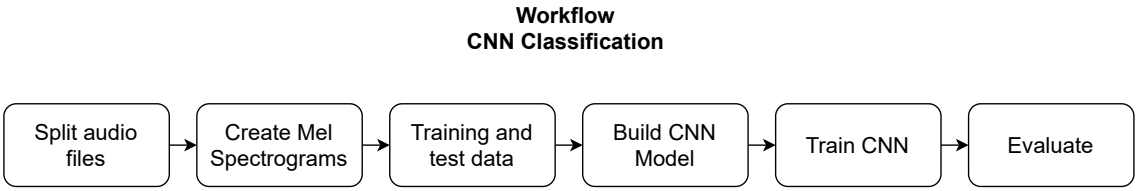


Figure 3: Flowchart of the Image-based Classification.

4.1 Preprocessing

The GTZAN dataset comes with labeled data and spectrograms for the full 30-second sample files. As mentioned in chapter **2. Dataset** the original dataset contains a total of 900 music samples, which is a bit small. A song in a specific genre might sound different depending on verse, chorus, outro, etc. It can therefore be assumed that splitting the data into 3-second samples should work as a data augmentation to generate more training data. Each 3-second sample should once again be stored in its corresponding genre directory. In total, this would increase the dataset size to 9000, meaning 1000 samples for each genre.

4.1.1 Create Mel Spectrograms

Since this method is based on Image classification, the sample music files need to be visually represented. spectrograms are a visual way of representing how the frequency of a signal varies over time [6]. Figure 4 shows a spectrogram example, where the frequency is represented on the y-axis, time in seconds on the x-axis and the color intensity represents the intensity in decibels.

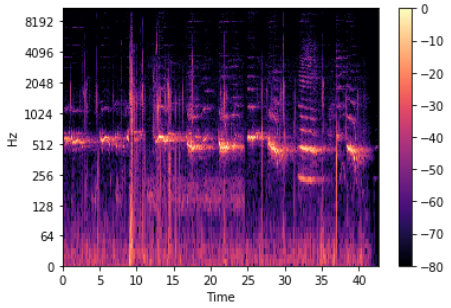


Figure 4: Example of a spectrogram with labeled axes. [1]

4.1.2 Data Generator

Training and validation input data for our Neural NetWork will be generated using the Keras ImageDataGenerator class[2]. It is usually used to simplify training on large dataset, or for data augmentation but works well for smaller datasets as well. The generator also includes scaling which is needed as the original images contain RGB coefficients in the range of 0-255. For the typical learning rate, such values are too high for the model to process. By using a scaling factor of 1/255 we should get values between 0-1.

4.2 Building the Model

Now that we have created the mel spectrograms in the preprocessing, we can build the actual model that we are going to use, and the algorithm that we have chosen for this method is the convolutional neural network (CNN).

4.2.1 Convolution Neural Network

CNN is a deep learning algorithm that takes in an input image and assigns importance to various aspects/objects in the image and learns to be able to differentiate one from another. CNN is optimized for processing data with a grid-like topology, e.g., an image [3]. Each pixel of an image represents a binary value that could be translated into colors, intensity, etc. Images are therefore very well suited as an input for CNN. Since spectrograms of a song are kind of like an image, each with its distinct patterns, it makes perfect sense to use CNN for this method.

4.2.2 The Model

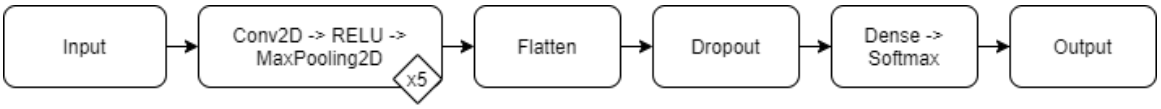


Figure 5: CNN Model

The model's workflow begins by passing the input spectrogram through the CNN layers and then flattening their output. It then passes the output after flattening through the dropout layer and then lastly to the dense layer with softmax activation to perform classification as shown in figure 5.

The convolutional block of the model consists of a 2D convolutional layer followed by a 2D max-pooling layer. There are a total of five blocks of convolutional max-pooling layers and their final output are flattened, and a dropout layer exists to reduce any overfitting that

[2]https://www.tensorflow.org/api$_d ocs/python/tf/keras/preprocessing/image/ImageDataGenerator$

would occur.

4.3 Training CNN

Once the CNN model has been defined we need to select what type of input is best suited for our project. The shape of a CNN model is a 4D array where we defined batch_size, height, width, and the depth of the model. Classes are another input that is necessary for the CNN model, classes define the amounts of labels that are going to be used when training the model. In this project, we used 9 different types of music genres. The Adam was used as an optimization algorithm due to it being able to handle sparse gradients on noisy problems. The learning rate in the Adam algorithm is set to 0.0003. Once the model has been defined and the learning rate set we compile and run the GenreModel.

A model checkpoint function is implemented as a means to save the model or weights once the training reaches a certain interval. If the training is interrupted for any reason we can continue the run from the latest checkpoint and do not need to start from the beginning. An earlystopping function is implemented. EarlyStopping is used to monitor a certain metric, once the monitored metric no longer is improving the training stops. In our project, the monitored metric is "loss" and the mode "min". During the fitting of the model, the training loop will at the end of each epoch check if the loss has decreased or not. The input patience is set to 10, which is the number of epochs that will be checked with no improvement before the training is stopped.

4.4 Evaluation

To evaluate the model, Keras built-in function Model.evaluate() is used. The function can simply use the datasets generated by Keras ImageDataGenerator. The Evaluation metric that will be used is the accuracy together with the loss. The model training history is also stored to produce a plot of the training progress. The plot containing loss and accuracy for both training and validation data should be evaluated to see if the training seems to have converged. If it has converged this would strengthen the reliability of the final result.

5 Result - Feature-based Classification

5.1 K-Nearest Neighbour

An accuracy of $\sim 75\%$ was achieved using the KNN model. The following confusion matrix was obtained using the KNN-model on the GTZAN dataset

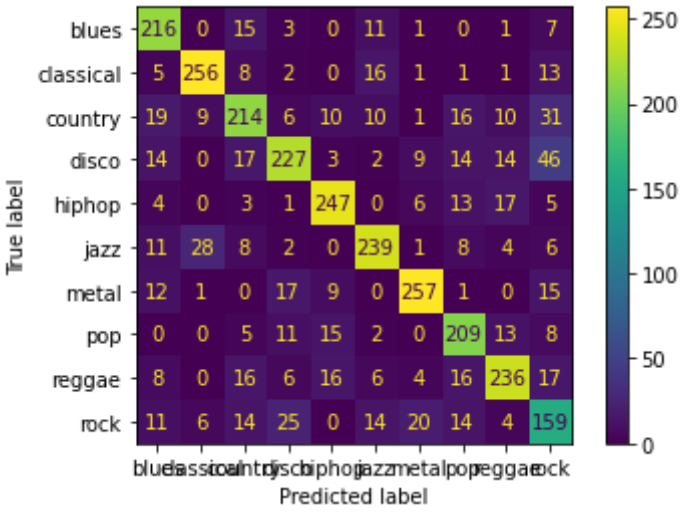


Figure 6: KNN confusion matrix. Achieved an accuracy of 0.75358.

as can be noted, the yellow color signifies a high number of classifications while purple is a low number of classifications. Furthermore, the diagonal is a correctly classified genre while other squares are incorrectly classified. The easiest genres to classify were classical, metal, and hip hop while the hardest genres to classify were rock, pop, blues and country.

Common misclassifications were that rock was often misclassified as disco or as metal, jazz was often misclassified as classical music, and country as rock.

5.2 Support Vector Machine

An accuracy of $\sim 80\%$ was achieved using the SVM model. The following confusion matrix was obtained using the KNN-model on the GTZAN dataset

The confusion matrix of the SVM shows similar results to the confusion matrix of the KNN as classical and metal were still the easiest to classify. Rock is also still the hardest genre to classify according to this confusion matrix. Similar to KNN, the SVM classifier often confused rock and disco music, as well as classical and jazz music. Though SVM was better at distinguishing rock and metal, it often misclassified disco as hip hop, which was not a problem for KNN. However, the results are slightly better in terms of total accuracy using the SVM model.

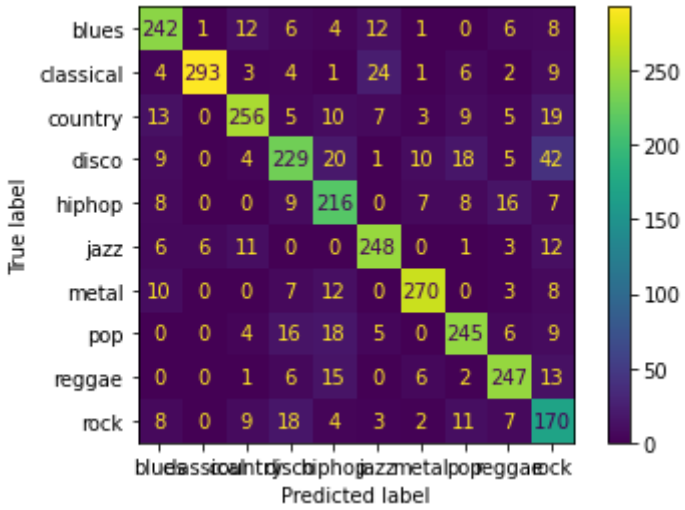


Figure 7: SVM confusion matrix. Achieved an accuracy of 0.806139.

5.3 Feature Analysis

5.3.1 PCA

After reducing the dimensions from 57 to 2, the data points of the different genres are quite distinguishable but close to each other as can be viewed in figure 8. The two-component PCA gave us information as to how different the genres are distinguishable in regards to the features. One can see that the data belonging to the classical (magenta dots) and rock (brown dots) genres are concentrated and distinct clusters, indicating that these genres will be rather easy to classify. Genres that overlap with each other, such as hip hop (dark blue), reggae (light blue) and pop (green), might be more difficult to tell apart. We also observed these trends in the confusion matrices, as mentioned in the previous sections. The rock data (yellow points) does not stand out in the scatterplot, which means that the data has a large variance and overlaps with many of the other genres. Hence our classifiers had the lowest accuracy for the rock genre.

5.3.2 Feature Importance

The top 10 most important features as extracted from the LDA and random forest classifier. It is worth noting that the ranking of these features differ from run to run, but these always appear in the top 10 (in our runs).

1. ms_mean
2. $spectral\_bandwidth\_mean$
3. $rolloff\_mean$

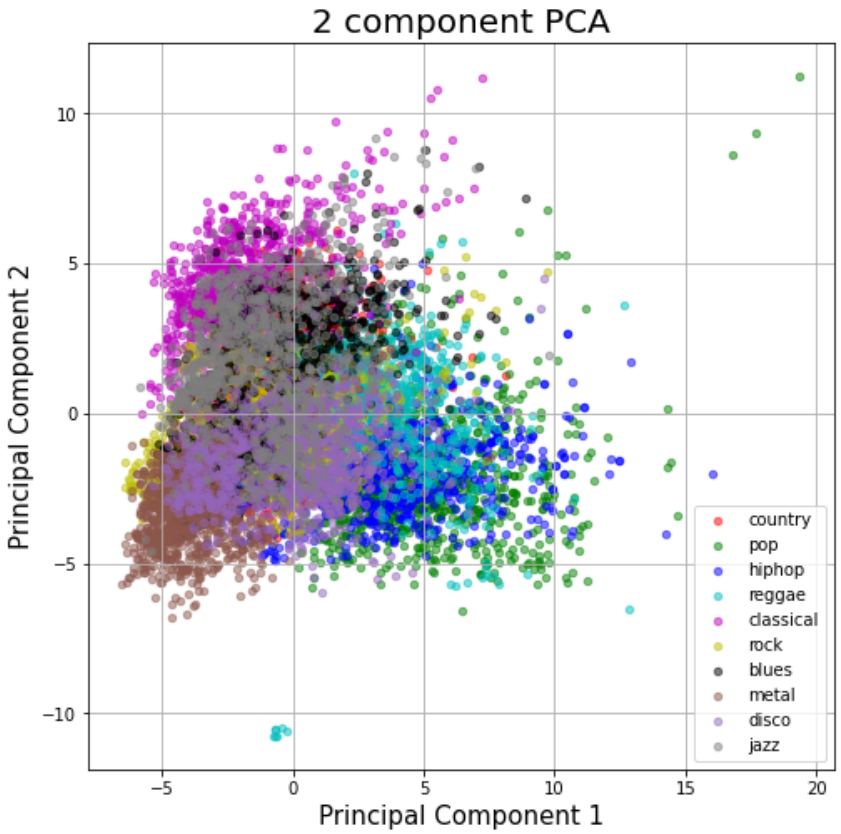
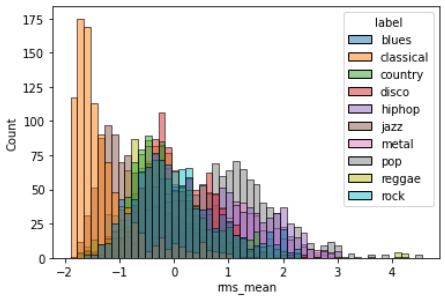


Figure 8: Scatterplot of the PCA transformed data

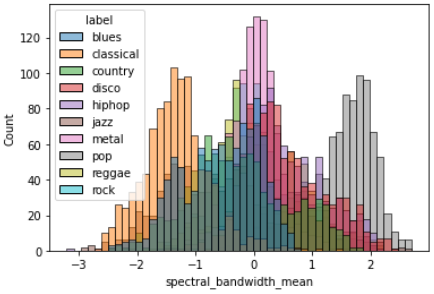
4. $mfcc1\_mean$
5. $chroma\_stft\_mean$
6. $perceptr\_var$
7. $spectral\_centroid\_mean$
8. $harmony\_var$
9. $mfcc4\_mean$
10. $rolloff\_var$

The effects on the prediction of the genre can be viewed in the histograms in figure 9, where the x-axis represents the value of the feature and the y-axis represents the number of data points from the same genre that were recorded with the value of the current feature. The colors represent the genre.

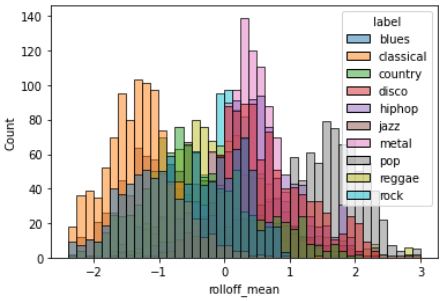
From the histograms, we can observe that the classical genre (orange color) has a very distinct peak for almost all figures. This is the case for the means and for the variances of features. Hence classical music was the easiest to classify. Other genres that seem to have unique characteristics are metal (pink color) and pop (grey color).



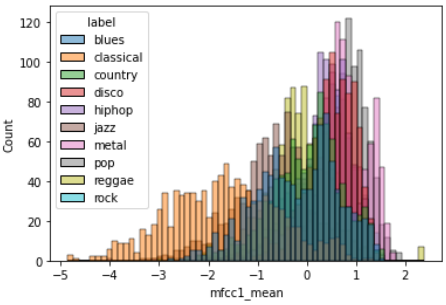
(a) ms_mean



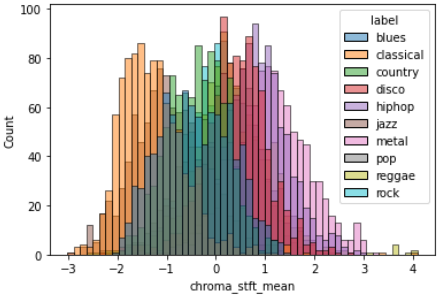
(b) $spectral\_bandwidth\_mean$



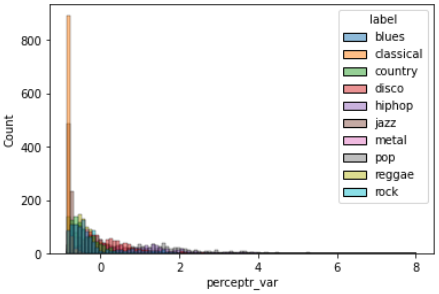
(c) $rolloff\_mean$



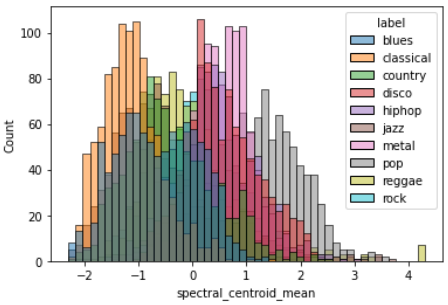
(d) $mfcc1\_mean$



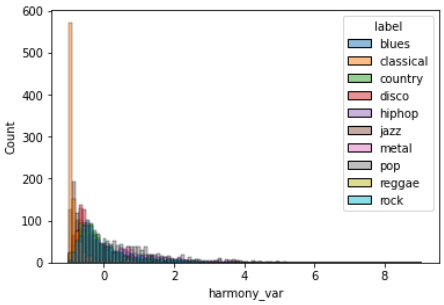
(e) $chroma\_stft\_mean$



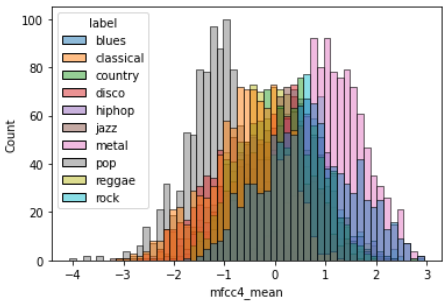
(f) $perceptr\_var$



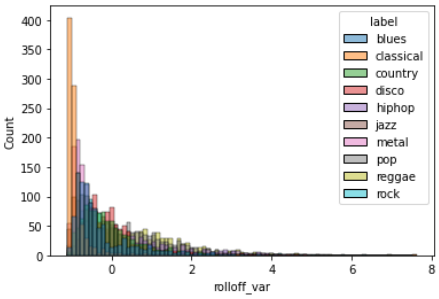
(g) *spectral_centroid_mean*



(h) *harmony_var*



(i) $mfcc4\_mean$



(j) $rolloff\_var$

Figure 9: Distribution of the 10 most important features

6 Result - Image-based Classification on Spectograms

The Image-based classification model trained for 37 epochs to reach its higher validation accuracy. The model trained for another 10 epoch but the Early stopping was triggered at epoch 47 as there were no improvements in terms of accuracy.

Both the training and validation data were evaluated as shown in figure 6 and figure 7 respectively. The training data had an accuracy of 99.09% while the validation set had an accuracy of 84.44%.

```
model.evaluate(train_generator)
```

```
64/64 [==============================] - 1733s 27s/step - loss: 0.0404 - accuracy: 0.9909
[0.040374889969825745, 0.9908642172813416]
```

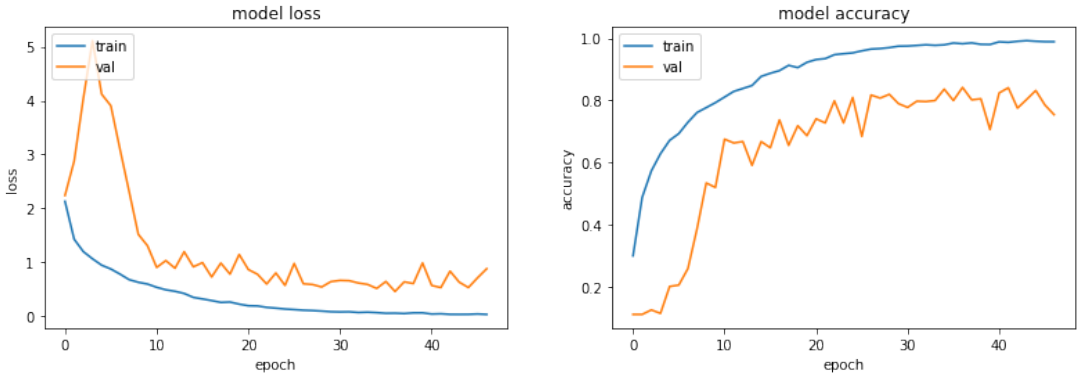
Figure 10: Evaluation of the training dataset.

The model loss is decreasing with increased accuracy. After 10 epochs the accuracy and loss for the validation data are slowly converging.

```
model.evaluate(vali_generator)
```

```
8/8 [==============================] - 150s 20s/step - loss: 0.6308 - accuracy: 0.8444
[0.6308470368385315, 0.8444444537162781]
```

Figure 11: Evaluation of the validation dataset.



(a) Model Loss

(b) Model Accuracy

Figure 12: CNN Model

7 Discussion

7.1 Feature-based Classification

The feature-based classification reached an accuracy of just above 80% using a Support Vector Machine. This seems reasonable, given that we only use spectral and rhythm features to predict the genres of songs. As described in section 5.3, we analyzed the features in regards to their ability to distinguish songs of different genres. We showed that it is possible to see some differences between the classes, but that it varies a lot for each genre. For instance, the feature analysis showed that genres such as jazz and metal had more distinct features than songs that belonged to the rock genre, indicating that some genres are more difficult to classify than others.

This poses the questions of what defines a genre: usually one would not use spectral features to describe what pop music is. Hence looking at additional data, that is not purely feature-based, might achieve better results. Another possible improvement could be to add weights to the classification in order to achieve more similar results for all genres, i.e. put a higher weight on the genres that are harder to classify to improve their accuracy and vice versa. But given the feature-based data, the SVM model achieved a good accuracy and we reason that the there will not be notable improvements by further tweaking the model or adding classification weights.

7.2 Image-based Classification on Spectograms

The Image-based Classification model was able to predict genres with a reasonably high accuracy of 84.4% using the validation data. The accuracy of the training data was at 99%, which would imply that the training have learned correctly. As the accuracy of the validation data was good, it should prove that the model did not overfit even though the training data were almost at 100%.

Looking at the training history graphs in figure 8, the training seems to have converged. There are still some spikes in the validation accuracy and loss, but there were no improvements in the last 10 epochs. Most likely there would not have been any significant improvements to increase the Early stopping patience.

7.3 General Discussion

Both proposed methods show promising results that could be compared to human-like accuracy. As discussed in *section 7.1* some genres are more prone to misclassification as their features are not completely distinguished from other genres. The problem with genre classification is that it is subjective. In our work, we focused on 10 general genres which only cover a small portion of all the genres out there. While there is no exact answer to how

many genres exists, the website MusicGenreList[3] suggest that there are 41 primary genres and 337 subgenres of music. Exactly which genre or subgenre each song corresponds to is not easy to define. It is usually in the genre that the majority of people would categorize it in.

Some sources of error in the evaluation can be a faulty dataset. While GTZAN is one of the most popular datasets for music genre classification, a study by Sturm [11] shows that 10.8% of the data is mislabeled. However, the evidence is based on musicological indications of genres, and how songs have been labelled by last.fm[4] users. Once again, partly based on the subjectivity of the consumers.

Overall, the complexity and subjective factors of genre classification show that we have trained models that classify genres based on generalized features. Therefore, a higher accuracy would not necessarily translate into a better model. Such a model might just be overfitted or biased to the dataset.

[3]https://www.musicgenreslist.com
[4]https://www.last.fm

8 Future Work

Future work might include expanding the methodologies trying a text-based classification which includes the lyrics of songs. The method of using the lyrics in the learning process may yield different results such as being able to differentiate between the different song genres. As discovered, some song genres are difficult to differentiate from one another because of how similar they are in song features (melody, pitch, etc.). However, a songs lyrics may differ more compared to song feature, making it possible to differentiate the genres that were difficult with the methodologies the used in this work.

As mentioned in the discussion, the study by Sturm [11] stated that roughly 10% of the data in the GTZAN data set were mislabeled, however, this were most likely due to a subjective labeling. It would still be interesting to know how the methodologies used in this work would fare and react against other data sets.

Finally, incorporating both methodologies into a single classification might increase the results even more. Instead of only analyzing and learning a songs features or spectrograms, having both in a single system may help in differentiating the genres that were difficult to differentiating with only one of the classifications. Also by including a text-based classification may even yield the ability to classify sub-genres lie pop-rock, J-pop, K-pop, etc.

9 Contributions

In the start, every member contributed to the project by reading research papers and articles on different projects in order to give us an idea on what we wanted to do. When the project was defined, the members did research on classification methods which could suit the project and data sets we would use. The work in this project was then divided up in two groups, with 3 persons in each group, one which did image-based classification and one which did feature-based classification. We worked closely together by having frequent zoom meetings, using a lot of Mob programming sessions.

9.1 Daniel Hu

Worked on the Image-based Classification. Worked mostly together with Drinas in the preprocessing of the GTZAN data. He focused mainly on generating the mel spectograms. It also included implementing a checkpoint function as the generation of spectograms were extremely slow, which made Google Colab time-out.

For the report Daniel focused on section 4.2 and 4.4 in the method which mainly included the CNN model. He also wrote the section 8 future work.

9.2 David Ångström

Worked on the feature-based classification. Started up by writing code for reading in the data set into Colab. Worked on the KNN implementation together with Lisa and Olivia, Worked on optimizing the the KNN and SVM classification methods to get higher accuracy values. Worked on feature importance together with Lisa. Focused on delimitations, SVM, PCA and Feature Importance in the report.

9.3 Drinas Kastrati

Worked on the Image-based Classification. Worked mainly together with Daniel on the preprocessing part of the implementation. His main responsibility was to split all the 30s songs into 3s segments and create the folder structure to hold all the labelled songs. He was also responsible for gathering the results from the training and evaluation. Drinas also helped Emil with the training of the model as we tried different architectures.

For the report, Drinas focused on section 4.3 Evaluation and collecting and writing the section 6. Results for the Image-based Classification.

9.4 Emil Bråkenhielm

Worked on the Image-based Classification. While most of the work was done together, Emil had the main responsibility of building the CNN architecture, implementing the callbacks

(checkpoint and early stopping) and training the CNN.

In the report he worked with section 2. Dataset, 4. Method 2 introduction followed by section 4.1 with its subsections. For the discussion he wrote the Image-based Classification as well as the general discussion 7.3.

9.5 Lisa Spahn Lundgren

Worked on the feature-based classification. Started working with Olivia to analyse and preprocess the data set, and dividing it up into training and testing sets. Worked on implementing the KNN, together with Olivia and David. Worked on feature importance together with David. Implemented a lot of the visualizations of the different results, in particular the feature histograms. As for the report, wrote parts of the method and the results for feature-based classification, and wrote the discussion for feature-based classification.

9.6 Olivia Shamon

Worked on the feature-based classification. Started up working with Lisa by analysing and preprocessing the data set, and dividing it up into training and testing data. Worked on implementing the KNN togther with Lisa and David. Implemented the SVM model which then later was optimized by David. As for the report, defined the basic structure in overleaf, wrote the introduction section for the report and also was part of writing the method for feature based classification.

References

[1] Hervé Abdi and Lynne J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459, 2010.

[2] Haneen Arafat Abu Alfeilat, Ahmad BA Hassanat, Omar Lasassmeh, Ahmad S Tarawneh, Mahmoud Bashir Alhasanat, Hamzeh S Eyal Salman, and VB Surya Prasath. Effects of distance measure choice on k-nearest neighbor classifier performance: a review. *Big data*, 7(4):221–248, 2019.

[3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[4] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18(1998):1–8, 1998.

[5] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[6] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. *arXiv preprint arXiv:1802.09697*, 2018.

[7] De Rosal Ignatius Moses Setiadi, Dewangga Satriya Rahardwika, Eko Hari Rachmawanto, Christy Atika Sari, Candra Irawan, Desi Purwanti Kusumaningrum, Nuri, and Swapaka Listya Trusthi. Comparison of svm, knn, and nb classifier for genre music classification based on metadata. *2020 International Seminar on Application for Technology of Information and Communication (iSemantic), Application for Technology of Information and Communication (iSemantic), 2020 International Seminar on*, pages 12 – 16, 2020.

[8] Ismail Bin Mohamad and Dauda Usman. Standardization and its effects on k-means clustering algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, 6(17):3299–3303, 2013.

[9] Ndiatenda Ndou, Ritesh Ajoodha, and Ashwini Jadhav. Music genre classification: A review of deep-learning and traditional machine-learning approaches. In *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–6, 2021.

[10] Iman Paryudi. What affects k value selection in k-nearest neighbor. *International Journal of Scientific & Technology Research*, 8:86–92, 2019.

[11] Bob L Sturm. An analysis of the gtzan music genre dataset. In *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, pages 7–12, 2012.

[12] George Tzanetakis, Georg Essl, and Perry Cook. Automatic musical genre classification of audio signals, 2001.