Benchmarking of State-of-the-Art Machine Learning Algorithms for Music Genre Classification

André Torvestad
Cybernetics and Robotics
NTNU
Trondheim, Norway
andre.torvestad@estudiantat.upc.edu

Frédérique Koopman
3mE - Technical Medicine
Delft University of Technology
Delft, Netherlands
frederique.koopman@estudiantat.upc.edu

Siri Rüegg
MSc Mechanical Engineering
ETH Zurich
Zurich, Switzerland
siri.leane.ruegg@estudiantat.upc.edu

***Abstract*—Music genre labels are useful for organizing songs, albums, and artists into larger groups that share similar musical characteristics. In this project, the most popular and state-of-the-art machine learning classifiers and algorithms are compared and tested for their suitability for music genre prediction. This work covers the preprocessing, the classification and the evaluation of the data. The focus is investigating the performance of common classifiers, the feedforward neural network and the convolutional neural network. Finally, a qualitative analysis of the results sheds some light on the behavior of the different algorithms on the single-label classification task.**

***Index Terms*—Machine Learning, Music Analysis, Genre Classification, Multi-Class Classification, Deep Learning, Single-Label Classification**

I. Introduction

In this project, data is retrieved from the GTZAN dataset [1] regarding music genre classification. This dataset consist of 100 audio files for each genre. 10 genres are incorporated in this dataset. Two files are provided within the dataset: a file with audio fragments of each 30 seconds and a file with audio fragments of each 3 seconds. In the 3 second file, the 30 second audio fragments are cut in smaller fragments of 3 seconds each, which yields 10 times more data. Because it is preferred to have more data, in this project the 3 second file is used. The audio fragments are described by 57 acoustic features, like the spectral centroid, spectral rolloff and Mel frequency cepstral coefficients. The aim of this project is to perform a benchmarking of machine learning algorithms and neural networks to evaluate their performance on music genre classification. In this report, the different steps taken and the rationale for them are explained and justified.

II. State of the Art

To date, the GTZAN dataset has been widely used in different approaches to music genre classification. For example, Feng et al. [2] built a hybrid architecture consisting of paralleling Convolutional Neural Network and Bidirectional Recurrent Neural Network Blocks to classify music genre from spectrograms. Using 1 layer for the RNN, they reached an accuracy of up to 90.2% for the genre classification.

Agrawal [3] used MFCC (Mel-frequency cepstral coefficients) to extract features from the GTZAN dataset. After this, they built an KNN classifier from scratch that finds K number of nearest neighbours based on the extracted features. They reached and accuracy of 70% with this.

Benchmarking machine learning algorithms for a specific dataset is done on a regular basis to evaluate which algorithm works best such as Xiao et al. [4] that were benchmarking machine learning algorithms for the fashion-MNIST dataset. Another similar approach was done by Xie et at. that aimed to compare the performance of several commonly known machine-learning (ML) models versus a classic Autoregression with Exogenous inputs (ARX) model in the prediction of blood glucose (BG) levels using time-series data of patients with Type 1 diabetes (T1D) [5].

III. Proposed Approach

The proposed approach consists of four different steps:

- Data exploration
- Data preprocessing
- Classical machine learning algorithms, feed-forward neural networks and convolutional neural networks
- Evaluation and validation

In order to understand the data from the GTZAN dataset, it will be explored and an overview over the dataset will be given. Missing data will be detected and a t-SNE (T-distributed Stochastic Neighbor Embedding) projection will be used to visualize the high-dimensional data. A PCA (principal component analysis) will be accomplished to elaborate how much the dimensionality of the dataset could be reduced. Then, features that don't hold relevant information for the classifier - such as the song title - will be cleaned from the dataset. In addition, the data will be standardized since the features have varying means and ranges. The data is thereafter split into training and test sets to be used in the classification processes.

In the first part the data is used to fit some of the most common classifiers, as specified in section IV-C. These classifiers will be compared and evaluated. To achieve better accuracies and scores, hyperparameter tuning algorithms will be used.

Neural networks are at the basis of the advancements made in artificial intelligence today, which makes them important to understand. In order to explain how they work and what parameters affect their behavior, a feed-forward neural network is built to classify the song samples. Different structures of

neural networks are tried out in order to experiment with how the parameters affect the complexity of the network and how the complexity affects the classification accuracy. As stated by the No Free Lunch Theorem [6], every problem has it's own optimal solution. The neural network best fitted to the problem at hand will be proposed in the end.

Finally, as also images (Mel Spectrograms) representing each audio file are available in the dataset, a Convolutional Neural Network would be an interesting Deep Learning algorithm to test the predictability of the music genre based on the spectrogram. A CNN learns to differentiate various aspects within an image from each other and to assign importance to them in the determination of the class [7]. Furthermore, a CNN can capture spatial and temporal dependencies in an image, which is of high importance for this project as a multi-class problem is proposed.

IV. Experimentation

A. Data Exploration

First of all, it is of high importance to get a global idea of the data set, the features it contains and potential missing values. As mentioned in section I, the data set chosen for this project consists of 100 audio fragments for each genre, all split in 10 smaller fragments. This results in a total of 9.990 "samples", as one song is missing. The data set contains 60 features, describing different aspects of the audio fragments. The data is checked for missing values, but doesn't contain any. Therefore, missing data imputation is not required for this data set. Furthermore, a t-SNE projection was computed since t-SNE can be very helpful to visualize the high-dimensional data. This was done by using the `TSNE` function from the `sklearn.manifold` module. As can be seen from the t-SNE projection in Figure 1, some genres are better distinguishable based on their features than others. For example, classical music and pop music can be well categorized, whereas the blues or disco are much more "all over the place".

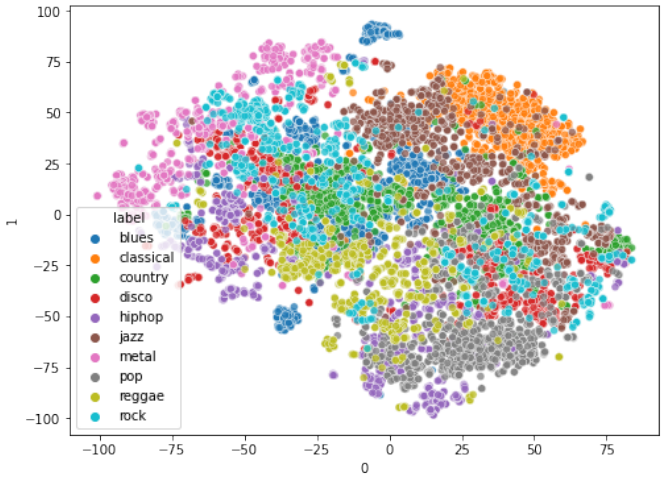


Fig. 1. t-SNE Projection of the Song Data

To understand how much the data could be reduced, a PCA analysis was performed. For this the `PCA` function from the `sklearn.decomposition` module was used. From Figure 2 it is evident that 38 linear combinations of the features, so-called principle components, are necessary to describe at least 95% of the variance of the data set.

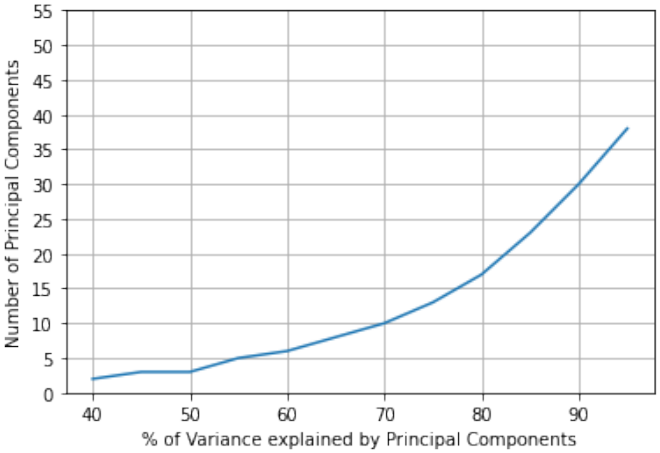
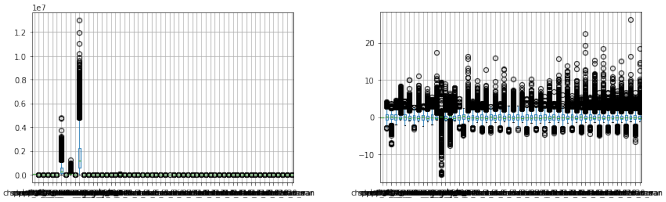


Fig. 2. % of Variance explained by Principal Components

B. Data Preprocessing

When visualizing the different features, two features are deleted from the data as they do not contain any valuable information. First, the file name can be removed as it tells nothing about the data except for the genre, which besides is already a separate feature. Also, the length of the samples is not relevant for the classification, since the length is similar for all samples.

As the features have different mean values and ranges, standardization must be performed to avoid bias in the data. If no standardization is done, certain features might carry more weight than others. The `StandardScaler` function from the `sklearn.preprocessing` module is used for standardization, as this is the specified standardization method for deep learning algorithms. In Figure 3, the boxplot of the data can be seen before and after the standardisation. It is obvious that the mean values of the various features had an immense offset and therefore a standardisation holds great advantage.



(a) Before Normalization (b) After Normalization

Fig. 3. Normalization of Input Data

After the standardisation, the data is split in training and test data. For this, a train-test split of 70/30 was used as this is very common and recommended in machine learning. It was done using the `train_test_split` function from the `sklearn.model_selection` module.

C. Common Classifiers

As the data has been preprocessed and is now ready for training, eight different types of classifiers from the `sklearn` library were used to classify the data. All classifiers were first run in a configuration with default parameters. The classifiers used are summarized below:

- K-Nearest Neighbors Classifier (kNN)
- Random Forest Classifier (RFC)
- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- AdaBoost Classifier
- Multi-layer Perceptron Classifier (MLP)
- Decision Tree Classifier (DT)
- Support Vector Machine Classifier (SVM)

From Figure 4 it is visible, that the k-nearest neighbour classifier had the highest accuracy with an accuracy of 0.91. The multi-layer perceptron classifier performed high as well with an accuracy of 0.86. It is noticeable that AdaBoost had a very bad performance. After investigating this classifier, it was apparent that this classifier is specially built for binary classification problems which is not the case in this project and therefore it makes sense that the performance of this classifier for the given data set is rather low.

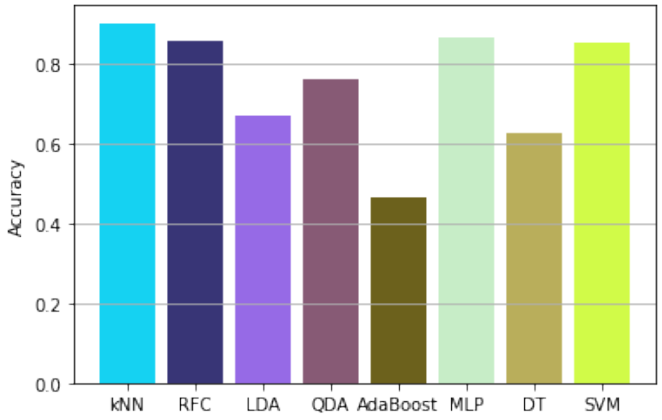


Fig. 4. Accuracy Comparison of 8 Common Classifiers

To determine if the kNN classifier can have an even higher accuracy, the classifier was run 20 times with increasing number of neighbours. However, when looking at Figure 5, it is clear that for this data set, an increasing number of neighbors usually reduces the accuracy.

Another possibility to increase the accuracy even more is using a hyperparameter tuning algorithm. The algorithm used to tune the hyperparametes is called `RandomizedSearchCV` from the `sklearn.model_selection` module. This function generates a random set of input parameters within a predefined range of values and then selects the one with the highest accuracy.

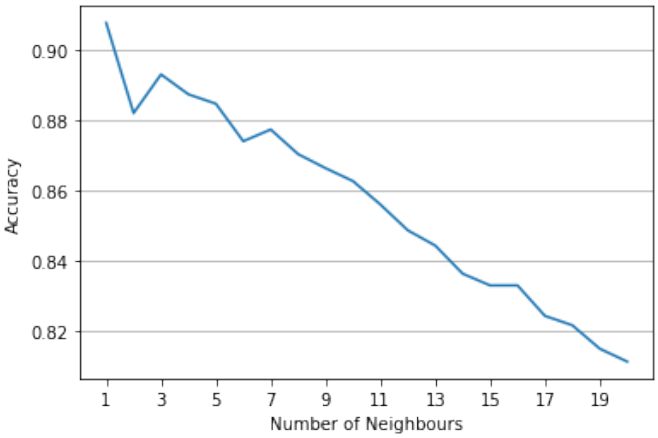


Fig. 5. Determination of best Number of Neighbours

This algorithm was tested for the two best performing algorithms in default state, namely kNN and MLP. For kNN the parameters leaf_size (Leaf size passed to BallTree or KDTree), p (power parameter for the minkowski metric) and the n_neighbors (Number of neighbors to use) were tuned. In total, 100 random configurations were tested. The configuration with the best result had `leaf_size: 44`, `p: 1` and `n_neighbors: 1`. This led to an increase of the accuracy of 0.02.

TABLE I
RESULTS KNN AFTER HYPERPARAMETER TUNING

	precision	recall	f1-score	support
blues	0.95	0.93	0.94	298
classical	0.95	0.98	0.96	318
country	0.89	0.87	0.88	311
disco	0.91	0.94	0.92	320
hiphop	0.95	0.96	0.96	302
jazz	0.93	0.93	0.93	280
metal	0.98	0.97	0.97	293
pop	0.96	0.88	0.92	304
reggae	0.90	0.97	0.93	295
rock	0.89	0.87	0.88	276
accuracy	0.93	2997	0.77	276
macro avg	0.93	0.93	0.93	2997
weighted avg	0.93	0.93	0.93	2997

For the MLP hyperparameter tuning the parameters activation (Activation function for the hidden layer), alpha (Strength of the L2 regularization term), hidden_layer_sizes (The i^{th} element represents the number of neurons in the i^{th} hidden layer), learning_rate (Learning rate schedule for weight updates) and solver (The solver for weight optimization) were tuned. In total 100 random configurations were tested. The configuration with the best result had `activation: 'relu'`, `alpha: 0.05`, `hidden_layer_sizes: (50, 100, 50)`, `learning_rate: 'adaptive'` and `solver: 'adam'`. This led to no increase of the accuracy.

TABLE II
RESULTS MLP AFTER HYPERPARAMETER TUNING

	precision	recall	f1-score	support
blues	0.87	0.88	0.87	298
classical	0.91	0.92	0.92	318
country	0.82	0.77	0.79	311
disco	0.81	0.88	0.84	320
hiphop	0.84	0.88	0.86	302
jazz	0.88	0.85	0.87	280
metal	0.93	0.90	0.91	293
pop	0.90	0.83	0.86	304
reggae	0.85	0.87	0.86	295
rock	0.76	0.78	0.77	276
accuracy	0.86			2997
macro avg	0.86	0.86	0.86	2997
weighted avg	0.86	0.86	0.86	2997

In some situations it can be important that the dimensionality of the data set is reduced. This decreases the time and storage space required for running the algorithm and also helps remove multi-colinearity, which in turn improves the interpretation of the parameters of the machine learning model. This is the reason why it was investigated how the accuracy behaves with the decrease of the number of features that were taken into account for the model. Therefore, an algorithm was needed to select the best features. This was done using the functions `SelectKBest` and `f_classif` from the module `sklearn.feature_selection`. In Figure 6 it can be seen, that with the 30 best features one has almost the same accuracy as with the complete data set of 58 features. This means that the data set can almost be reduced by half and still have an acceptable accuracy.

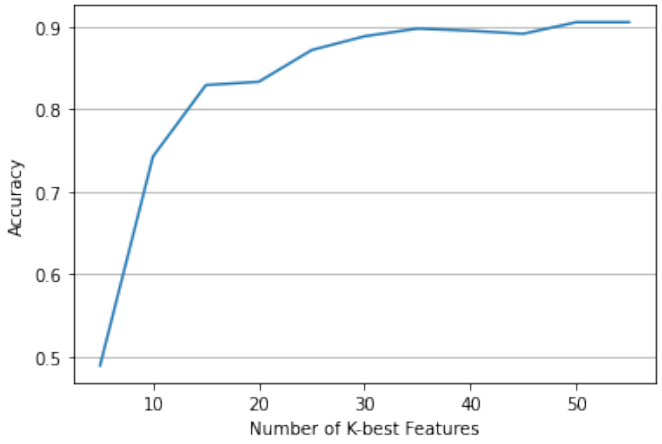


Fig. 6. Feature Selection by k-best features

D. Neural Network

The network built is the sequential model provided by Keras. The models use the Adam optimizer, which has the property of carefully choosing the step sizes to optimize it [8]. Furthermore, the models use the categorical cross-entropy loss function (as the problem proposed is multi_class) and the accuracy as the evaluation metric.

The baseframe of the neural network are the 57 input neurons corresponding to each feature and the 10 output neurons corresponding to each genre. Layers of neurons will be added in-between these layers in order to create the network, so-called hidden layers. More layers and more neurons complexify the network, as they increases the number of tunable parameters. The layers and their numbers of neurons are two of the parameters to be tuned in order to find a well-fitted structure for the given problem. Additionally, the number of epochs are tuned.

In order to evaluate whether the complexity of the model should increase or decrease, the loss function is evaluated. By using the `model.fit` function from Keras one can assess the model after each epoch at both a training and a validation set. In each epoch the model takes 20% of the training data and uses it to validate the model. By comparing the loss values from both the training and the validation set the model complexity can be assessed. If the loss from the training set goes towards being noticeably lower than from the validation set, the model is too complex. It overfits the training data so much that the model is not applicable for general data outside of the training set. If the loss function continues to decrease, however, the model is not complex enough to learn the data fast enough. The model is then underfitting.

The initial experiments are performed with shallow neural networks and from there the complexity is increased. This is done as the solution should be as simple as possible, rather than being too complex. As can be seen from figure 7 the network is underfitting, as expected. This model has one hidden layer with 57 neurons. The result is not bad, however, as the model achieves an accuracy of 79% on the testing data.

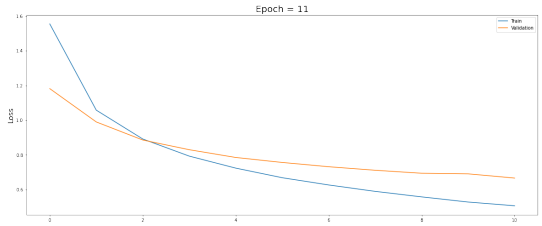


Fig. 7. Loss function - shallow network

More layers are added to increase the complexity. As seen in figure 8, adding two more layers - of 57 and 114 neurons - does not yield a lot better result. The resulting accuracy is 83%, a small step up from the previous result. However, this time the network appears to slightly overfit, as the loss for the training set continues to decrease, whereas it appears to have stabilised for the validation set.

As overfitting has occured, the model should not be more complex. In order to boost the behavior of the system, the number of epochs is tuned. Increasing the number of epochs results in the system overfitting even more, however, which must be prevented. An effective way of combating overfitting

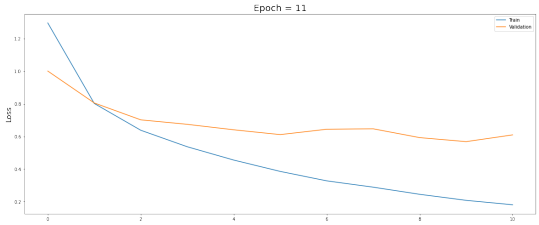


Fig. 8. Loss function - 3 hidden layers

is the introduction of dropouts. Dropout works by dropping the output from random neurons during training [9], which increases the robustness of the trained model. The same 3-layered network as in the previous paragraph is used, but with 50 epochs. The effect of introducing dropout is great, as can be seen in figure 9. Dropout changes the outcome of the model from clear overfitting to a well fit. It does not change the resulting accuracy of the model, however, as evaluation on the evaluation set yields 85% accuracy for both models. The overfitting model is discarded nonetheless, as it is an unwanted feature for a neural network.

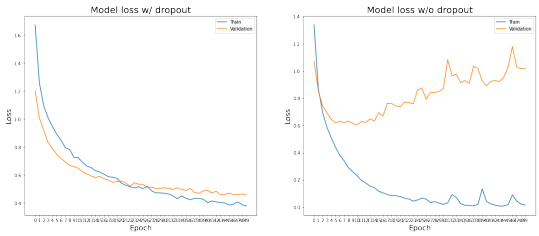


Fig. 9. Model loss with and without dropout

Through the experiments done in this subsection, it is apparent that the neural network should not be too complex and that dropout should be applied. From the art of tuning, based on the aforementioned knowledge acquired, a small increase in accuracy is achieved. This model is selected to be the best neural network for this specific genre classification. The model summary can be seen in figure 10. This network contains only two hidden layers with many neurons, applies dropout and has 32 epochs. The accuracy is increased to 88% - a good result on the data set. It does not perform as well as the kNN algorithm, but it outperforms the MLP with 2% better accuracy.

E. Convolutional Neural Network

Another approach is taken in this section, to see whether music genres can also be classified by a CNN based on the spectrograms. The input for the CNN are the spectrograms of the different genres. For each genre, 100 spectrograms are available. A common thing to do when building a deep neural network to classify images, is image augmentation. This is a process used to generate artificial data by random rotation, shifts, shear and flips [10]. Especially since 100 spectrograms per genre is not considered as a very large dataset, image augmentation can improve the performance of the classifier.

Layer (type)	Output Shape	Param #
dense_21 (Dense)	(None, 114)	6612
dropout_12 (Dropout)	(None, 114)	0
dense_22 (Dense)	(None, 171)	19665
dropout_13 (Dropout)	(None, 171)	0
dense_23 (Dense)	(None, 10)	1720

Total params: 27,997
Trainable params: 27,997
Non-trainable params: 0

Fig. 10. Optimal Neural Network

Furthermore, the images are rescaled in order to get all pixel values between 1 and 0.

A CNN is built with 3 convolutional layers, with increasing numbers of filters applied (32, 64, 128). The kernel size is 3 for each convolutional layer and a Rectified Linear Unit (ReLU) is used as an activation function, as this is the most commonly used activation function in CNN [11]. After each convolutional layer, a MaxPooling layer is added, with a pooling size of 2. Then, a Flatten Layer is inserted in order for the data to fit the Dense layer coming after this. Three Dense layers are used, the first one with 128 neurons, the second one with 64 neurons and the final one with 10 neurons (as this final layer has to be equal to the number of classes). For this final Dense layer, a SoftMax activation function is used, which is the preferred method for multi-classification problems. Finally, the loss function used is the categorical crossentropy, again, because of the multi-classification problem. This loss function summarizes the average difference between the actual and predicted probability distributions for all classes, and tries to minimize this value [12]. The Adam optimizer is used (Gradient Descent), which is the most commonly used optimizer in CNN and the evaluation metric used is the accuracy. In Figure 11, a summary of the construction of the CNN can be found.

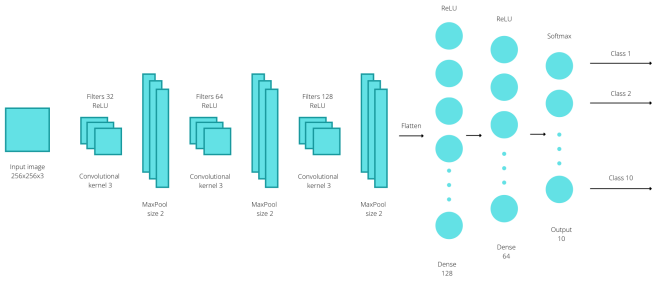


Fig. 11. The construction of the CNN

After building the network, the model has to be fitted. The model is trained on 70% of the data and validated with the remaining 30% of the data. Then, two parameters have to be specified: the batch size and the number of epochs. A batch size of 32 is chosen, as this is a good default value [13]. Different numbers of epochs have been tested in order to find the optimal solution. Finally, the Convolutional Neural Network is fitted and the accuracy and loss can be evaluated

after each epoch. Also, the learning curves are plotted for the training and validation data.

First, 30 epochs are used. In Figure 12, the learning curves are visualized. It can be seen that especially the validation loss and accuracy are really unpredictable with large outliers. A possible cause for this might be the number of epochs. Therefore, the number of epochs is increased to 100, as this could improve the stability of the validation and the classification accuracy.

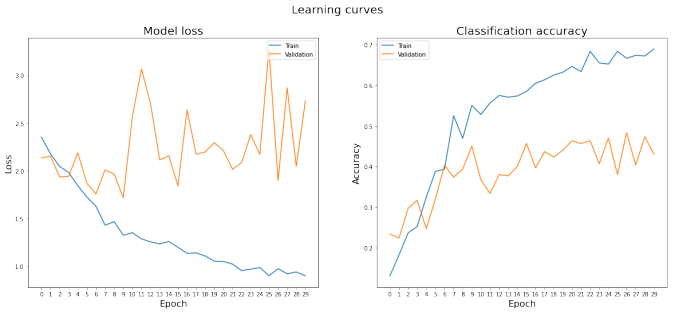


Fig. 12. Learning curves with 30 epochs

When evaluating the results from the CNN with 100 epochs (Figure 13), some interesting observations can be made. Despite of the training accuracy increasing throughout the training and ending around the 90%, the classification accuracy for the validation set remains around the 50%. This indicates that the model is not good at classifying the spectrograms. Several reasons can be thought of for this. First of all, possibly more data is required for the network to further learn. Despite the fact that image augmentation has been performed in order to generate more artificial data, the dataset remains rather ”small”. Another reason could be the complexity of the CNN. Adding more layers might improve the accuracy, as the classification might become more precise. On the other hand, it can also cause the model to become too complex, leading to an even higher computational time and possibly the occurence of overfitting.

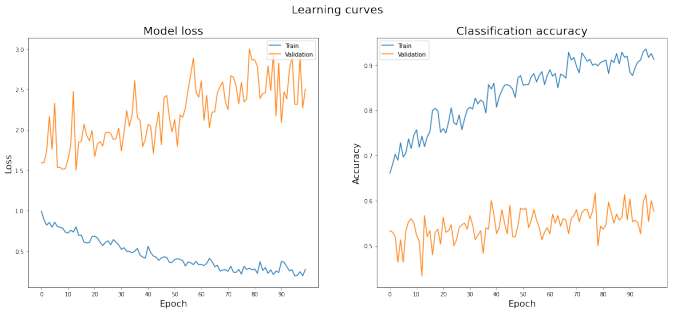


Fig. 13. Learning curves with 100 epochs

Lastly, it is important to note that a spectrogram is a heat map, which means that is a visual representation of intensities shown by colors or brightnesses. As every spectrogram from each different genre will more or less include the same colors/brightnesses (just in different ratios), it can be imagined that it is quite hard for the CNN to extract clear features from this that are highly specific for one genre (as all other genres will include this feature as well). This reason therefore contributes to the belief that a CNN for spectrograms might not be the most evident way of classifying music genres.

Another possible hypothesis is that the 10 genres together can not be distinguished well enough. However, if three genres are picked that are visually very different based on the spectrograms, the CNN might perform better. It is interesting to find out if a very simple CNN on ”simple data” (i.e. three genres only) will be able to classify the genres with a high accuracy. As the spectrograms, which are the input for the CNN, are representations of intensities and frequencies, image augmentation might mess up the temporal dependency of the pixels in the spectrograms. Therefore, to keep the data ”simple”, image augmentation is left out of the process this time. One spectrogram from each genre is randomly selected and plotted. These spectrograms have been visually evaluated and three genres are manually picked that differ most (Figure 14).

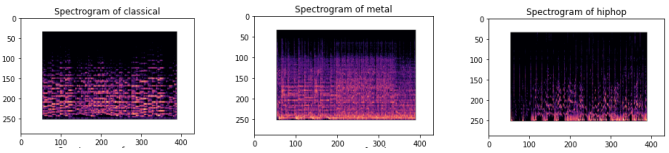


Fig. 14. Spectrograms of three different genres

Then, a very simple CNN (Figure 15) was built with one convolutional layer and one MaxPooling layer. After the flatten layer, one dense layer with 64 neurons was inserted. The final layer contains three neurons, are three the classification problem is now a three-class problem. The model was fitted on the data, and 15 epochs were used to prevent overfitting. In Figure 16, the learning curves are visualized. It can be concluded that the simple CNN on the ”simple” data works very well and reaches a validation accuracy of up to 92%.

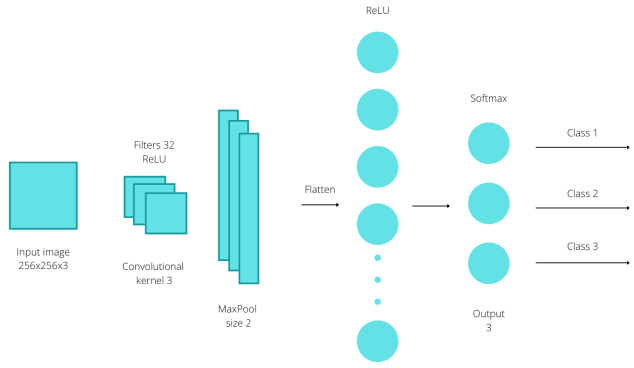


Fig. 15. The construction of the simple CNN

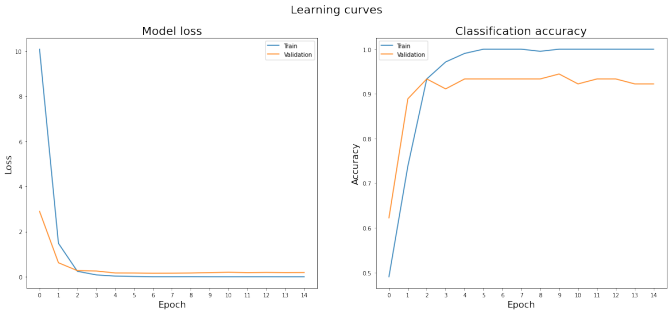


Fig. 16. Learning curves for a simple CNN on 3 genres

This section about Convolutional Neural Networks once again emphasizes that each machine learning algorithm performs different on different data. The optimal solution is not always obvious and requires data preprocessing and finetuning of hyperparameters. Especially for deep learning networks such as CNN's, a balance has to be found between complexity and simplicity, depending on the dataset proposed.

V. Conclusion

It is possible to perform genre classification with high accuracy on the selected data set - especially if using the extracted features. KNN stands out as the top performer, but also the MLP classifier from Keras and the neural network yield good results. The resulting accuracies is seen in figure 17.

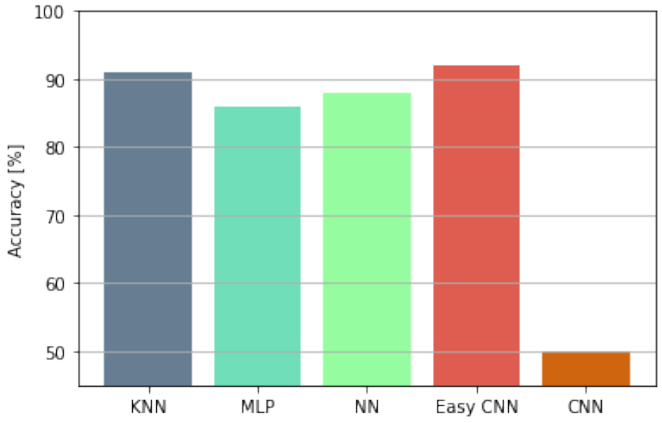


Fig. 17. Accuracy of the top performers (and the CNN)

The best results have come by using already extracted features from the data set, as done with the classical algorithms and the neural network. These features describe the data set well enough for the algorithms to classify the genres. Using a CNN to perform this feature extraction does not yield as good results, as the resulting classification is a lot worse. Having the knowledge about which features best describe the data is therefore better than allowing a neural network to find its own features. The performance is high when using histograms with clear distinctions, however.

The experimentation also shows that adding complexity does not yield any better results. This is especially highlighted with the KNN with 1 neighbor, one of the simplest machine learning algorithms, being the top performer for the data set. The same is shown when tuning the neural network, as a relatively shallow network results in the best accuracy. This benchmarking of solvers and parameters demonstrates a ground truth in pattern recognition and machine learning - there is no such thing as a free lunch.

References

[1] A. Olteanu. (2020) GTZAN Dataset - Music Genre Classification. [Online]. Available: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[2] L. Feng, S. Liu, and J. Yao, "Music genre classification with paralleling recurrent convolutional neural network," *arXiv preprint arXiv:1712.08370*, 2017.

[3] R. Agrawal. (2022) Music Genre Classification Project Using Machine Learning Techniques. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/03/music-genre-classification-project-using-machine-learning-techniques

[4] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[5] J. Xie and Q. Wang, "Benchmarking machine learning algorithms on blood glucose prediction for type i diabetes in comparison with classical time-series models," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 11, pp. 3101–3124, 2020.

[6] (2022) No free lunch theorem. [Online]. Available: https://en.wikipedia.org/wiki/No_free_lunch_theorem

[7] S. Saha. (2018) A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] D. Cochard. (2020) Prevent Overfitting Using Dropout. [Online]. Available: https://medium.com/axinc-ai/prevent-overfitting-using-dropout-53041b5a3797

[10] S. Lau. (2017) Image Augmentation for Deep Learning. [Online]. Available: https://towardsdatascience.com/image-augmentation-for-deep-learning-histogram-equalization-a71387f609b2

[11] V. Luhaniwal. (2019) Analyzing different types of activation functions in neural networks — which one to prefer? [Online]. Available: https://towardsdatascience.com/analyzing-different-types-of-activation-functions-in-neural-networks-which-one-to-prefer-e11649256209

[12] J. Brownlee. (2020) How to Choose Loss Functions When Training Deep Learning Neural Networks. [Online]. Available: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks

[13] J. Brownlee. (2019) How to Control the Stability of Training Neural Networks With the Batch Size. [Online]. Available: https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/