Genre Recommandation in Music: Harnessing Machine Learning with the GTZAN Dataset

Mălinescu Ionuț
Bachelor Student
Techical University of Cluj-Napoca
Cluj-Napoca, Romania

Prof.dr.ing. Mircea Giurgiu
Communications Department
Techical University of Cluj-Napoca
Cluj-Napoca, Romania

***Abstract*—This article explores several methods of Machine Learning (ML) that can be used for music genre classification, treating the methods as Proof of Concepts, with potential for future improvements. Using the GTZAN data set, which includes a total of 1000 audio tracks, split into 10 separate genres the ML models are trained and then evaluated. The two main types of architecture presented in the following paper are Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs). Both types of models present benefits and drawbacks, which are to be explored. Finally, a comparative approach is explored, in order to present the better option for the task of music classification, while also presenting possible future improvements.**

Keywords—music, categorization, machine learning, information extraction

I. INTRODUCTION

Almost one hundred years ago, music used to be shared mainly over a physical medium, mostly trough objects such as music vinyls that were engraved with groves, over which a fine needle would go and thus, resonate to recreate the sounds saved via this medium. Nowadays, the wide-spread development of the internet has lead to much easier ways of sharing any form of media, including music. In order to be able to facilitate a better way of navigating all the available media, labels had to be created to categorize similar music, labels known as "Genres".

These labels were mostly a subjective matter, and used according to the perception, the understanding of a song by a listener. Seeing as each human is unique, the ability to differentiate between genres is also unique to every listener. Factors such as genetics, age and hearing impairment may affect the perception of sound and the way, the brain transforms it into electrical signals in order for it to be processed. This matter makes the task of music classification, a difficult one, despite the great advancements in feature extraction from music.

This article attempts to present method of music classification according to genre, using feature extraction and machine learning concepts in combination with the GTZAN dataset to create a an automated model for genre detection, model which is further used to analyze music samples and create a genre-based recommendation system.

The article is organized as follows: Section 2. State of the art (Similar works to this paper), Section 3. The Proposed Method, Section 4. The Implementation, Section 5. Experimental Results and Section 6. Conclusions.

II. STATE OF THE ART

There already exist many papers and methodologies that treat the subject of music classification in combination with machine learning methods. One example is the paper by L. Moysis [1], where multiple deep learning approaches for feature extraction are presented and compared. These approaches are used for applications such as Music Recommendation, classification, prediction and identification systems. First architecture to be presented is using Fully Connected Deep Neural Networks (FCDNN), a type of network where each neuron in one layer has a direct connection to all the neurons in the subsequent layers. Such networks can prove to be exceptionally computationally demanding, depending on the number of layers involved. Despite this, such a method can also display exceptionally good results, in the case of this paper a precision score of 83% has been obtained, while making use of Spotify's Recsys Challenge 2018 million playlist dataset.

A different strategy devised by the author is using an Recurrent Neural Networks (RNN) architecture. Unlike FCDNN's, they have internal states (hidden states) which memorize information above previous inputs, making this architecture extremely advantageous in the case of sequential data. In the case of music classification, the RNN architecture, improved by attention mechanisms, which are used to assign different weights to the inputs, was able to score an accuracy of approximately 90%.

Another popular method for music classification is using Convolutional Neural Networks (CNN). Despite such an architecture being involved mostly for visual data tasks they were also successfully used in music classification tasks, as presented in the paper [1]. CNNs have the ability to adapt their spatial hierarchy based on the data input. Fine-tuning hyperparameters is a powerful tool for increasing accuracy of music genre detection tasks. The author present the results of their tests, using CNNs, for input data where two microphone arrangements were considered. The results reveal a high accuracy of almost 98%, using a large dataset from YouTube.

Alternative techniques presented in the paper include the usage of Generative Adversarial Networks (GAN). This architecture is composed of two networks: one generator and one discriminator. The generator has the task of generating realistic samples in order for them to be labeled by the discriminator as real, while the discriminator must differentiate between real and fake data. In an ideal case, the generator would create samples that would be impossible to differentiate from real ones. Conducting tests on multiple datasets, such as JSB Chorales, Maestro, Lakh MIDI, the recognition accuracy obtained is presented to be above 88% [1].

Considering the array of methods outlined by this paper, it can be inferred that there is no true method for genre detection. Any of those presented can be successfully used for detecting the genre of music, the accuracy being high enough in every case to properly detect it. In many cases, trying multiple methods can prove to be more beneficial than improving just one, as depending on the input data, one method could present better results than another possible one.

III. The proposed method

This article proposes the analysis of two methods for genre classification and thus for music recommendation. The outputs of both methods still have room for improvement so they should be taken in as Proof Of Concepts.

For the creation of this model, a multitude of acoustic features were systematically extracted from the audio signal to aid in the construction of a predictive model. The features range from spectral characteristics to frequency distributions, and were chosen to best fit the classification application involved in this method. The features extracted are the following:

i. **Mel-Frequency Cepstral Coefficients (MFCCs):** this coefficients are used to best characterize the short-term power spectrum of the sound; they summarize the spectral envelope of the signal in a minimal form; lower coefficients capture broader spectral features such as timbre or pitch while higher coefficients capture more detailed spectral structures like the Discrete Cosine Transform (DCT – used to decorrelate coefficients, thus reducing redundancy)

ii. **Spectral Rolloff:** this feature represents the precise frequency, below which more than 85% of the spectral energy is located; usually, a higher value of the Spectral Rolloff is associated with a brighter sound while a lower value may be associated with a darker, more noisier type of sound

iii. **Spectral Bandwidth:** if a feature that describes how the frequencies are distributed on the spectrum; higher values may indicate a wider frequency content, suggesting a richer and more complex or dynamic sound

iv. **Spectral Centroid:** represents the center of mass of the spectrum, representing the center of the frequency distribution; it can present the algorithm with information regarding the brightness of an audio signal; a higher spectral centroid suggests that the signal's energy is accumulated towards higher frequencies, meaning a more treble-oriented genre like Pop, Electronic Dance Music (EDM); a lower spectral centroid can indicate a darker more bass-oriented genre such as: Dubstep, Hip Hop, Trap

v. **Spectral Contrast:** it can give valuable information in regards to the distribution of frequencies across the audio spectrum; a broader bandwidth indicates a more wider range of frequency components, meaning a more complex and richer spectrum, while the opposite may determine a more limited one

vi. **Zero Crossing Rate (ZCR):** it is a measure of how often the audio signal changes its polarity, counting the number of times the signal crosses the x-axis; a higher ZCR value implies a more frequent and rapid change in amplitude, usually presenting a more percussive sound while more harmonic sounds present a lower ZCR value

vii. **Chroma Features:** represent the presence of different musical notes in the audio signal; they are computed using the Short-Time Fourier Transform (STFT) and correspond to the 12 pitch classes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B), each coefficient representing the energy of each one of the classes

For the training of the model a widely used dataset in music classification applications has been used, namely the GTZAN dataset. The datasets consists of a number of 1000 audio tracks each having an exact duration of 30 seconds adding up to almost 16 hours of music. The tracks cover a diverse range of musical genres including: blues, classical, country, disco, hip hop, jazz, metal, pop, reggae and rock. For each music genre, there are precisely 100 tracks allocated. The tracks are 22050Hz Mono 16-bit audio files in .waw format. Trough the 1000 tracks, the dataset presents great variability in terms of the content of its music, reflecting the real-world diversity of music, making it exceptionally suitable for music classification applications.

Using all the coefficients presented above in combination with the GTZAN dataset, a machine learning model has been developed to be used in automatic genre classification applications. The data has been split into 80% of it for the training of the algorithm while the other 20% has been used for its testing.

After trying multiple approaches, in order to present a comparative view of multiple machine learning (ML) algorithms, it has been decided that the most appropriate to be presented are the ones based on the following technologies: Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs).

Support Vector Machines are a type of supervised learning model that works by finding the hyperplane used to differentiate between the classes of the dataset. SVMs are very efficient because the functions that takes decision is defined only by one small subset of the training points, known as support vectors. The use of such an architecture is beneficial in high-dimensional spaces, as it can effectively find boundaries. This fact also presents the method with an inherent robustness to overfitting, which makes the algorithm really good at generalizing unforeseen data. This will be in easier to remark in the experimental part of this paper. It is also notable that SVMs focus on more significant data in order to take decisions, this brings two great benefits: the simplicity of the model, and thus the efficiency with which it operates in comparison to CNNs, as an example but also the high accuracy that can be obtained. This makes this method very practical for real-world applications.

On the other hand, Convolutional Neural Networks are a type of deep learning model vastly used in image processing tasks, but that can also be adapted to work with other forms of media, to automatically and adaptively learn hierarchies of features. A CNN network is composed of a multitude of layers such as convolutional layers, pooling layers and fully connected layers.

- **Convolutional Layers:** are used to apply a set of filters to the input data; by learning these filters, CNNs are able to extract data and create patterns from the initial data
- **Pooling Layers:** are used to reduce the dimensions of the feature maps created by convolutional layers and keep only the relevant information; such layers can help reduce the strain on the system by unnecessary data
- **Fully Connected Layers (Dense Layers):** are involved in for classification and regression tasks by taking the flattened output of the previous

CNNs are trained on a supervised approach, meaning that they are given a set of labeled data used for the training of the model. After training any of these models, the standard approach to determine the correctness of the model is to compute their accuracy, by taking into consideration how much of the data is correctly classified by the CNN.

Finally, after analyzing the accuracy of the models, a Python program is used to analyze a small number of sample songs and to classify them according to genre. The classification is done by giving a score to the genre with the greatest prevalence and by including also the ones that follow.

Using this data, a recommendation system may be implemented, by cycling the songs belonging to the same genre first and thus, that may have a similar characteristic and continuing with the ones that are less related to the original song.

IV. The implmentation

In the following section of the paper the implementation of the Python code, including the two architectures, namely the SVM and CNN, of the model will be presented in detail.

i. Feature Extraction

For both algorithms to be presented, the extraction of features has been done exactly in the same way, extracting the features presented above: MFCCs, Chroma Features, Spectral Rolloff, Spectral Bandwidth, Spectral Contrast, Zero Crossing Rate & Spectral Centroid.

TABLE I. Number of elements extracted per coefficient

Feature	***Number of coefficients extracted***
Zero – Crossing Rate (ZCR)	1
Spectral Centroid	1
Spectral Contrast	7*
Spectral Bandwidth	1
Spectral Rolloff	1
MFCC	13
Chroma Features	12
Total	36

[a.] The default value is 7.

Fig. 1. Coefficients extracted for each feature

Presented in the table above are all the coefficients extracted for each audio segment. Each one contributes with unique information about different properties of the audio signal, but by using a combination of all of them, an accurate classification model can be generated.

ii. SVC Model

The data is pre-loaded into an array alongside the labels of the audio tracks. It is further prepared to be introduced to the model.

The labels are first encoded, each receiving a unique integer value. The data is then split into two parts: one for training consisting of 80% of the total number of tracks and the other for testing, consisting of the rest of the audio tracks, 20%. This is done randomly, and the tracks chosen are saved so that testing may be reproduced if needed.

For the creation of the SVC model a pipeline is defined using the StandardScaler function imported from the scikit-learn machine learning Python library.

Hyperparameter tunning is performed for Grid Search, hyperparameters being variables that are not directly learned by the model, but which are implanted before the training process. Grid Search is performed with Cross-Validation, more precisely Stratifield k-fold cross-validation, with the parameter k set to 5, thus ensuring that every one of the 5 folds of the dataset preserves the percentage of samples from each class. Grid search takes the pipeline of the model (a set of data processing steps that are directly linked together) and performs an accuracy test on it. After this is done, the model with the best performance is extracted and kept; this model is the one trained with the best hyperparameters combination.

TABLE II. SVC – Hyperparameters

Hyperparameters	
Regularization Strength (C)	0.1
	1
	10
	1000
Kernel	Linear
	RBF

Fig. 2. Hyperparameters used for the creation of the SVC model

Figure 2 presents the hyperparameter combination tested in order to obtain as good of a model as possible. The regularization strength is a technique used for machine learning models to prevent overfitting. This phenomena refers to models which perform great on training data, but poorly on unknown data. This parameter is used to discourage very complex models by balancing the maximization of the margin (distance between the boundaries and the closest data) and the minimization of the classification error.

The kernel determines the type of kernel function to be used for the SVM algorithm. The kernel plays a crucial role in transforming the input data into a higher dimensional space, that can be later better analyzed by the model. The two types of kernels tested are the following:

- **Linear Kernel –** simplest type of kernel function; it computes the dot product between the original vectors directly in the original space (the decision boundary resulted trough this operation is a straight line)
- **Radial Basis Function (RBF) Kernel** – is a more flexible and powerful type of kernel function; it maps the input into a higher dimensional space using a Gaussian (Radial) space (the kernel is more flexible and can adapt to the size of the data)

After all the data is processed and the best model has been chosen, the model is saved under a .pkl format.

iii. CNN Model [2]

As presented above, for the CNN model the data is also pre-processed in the same way it is done for the SVC model. The same features are extracted, and the data is split yet again in the same way. Finally, an evaluation of the model is performed. The CNN architecture has been defined in this way, according to the paper [2]:

1. **Input Layer** – specifies the shape of the input data, expecting data of specific size
2. **Convolutional Layer I – "**(Conv2D(32,(3,3), padding=same))"; this defines a convolutional layers adding a number of 32 filters of 3x3 size each
3. **Convolutional Layer II – "**LeakReLU(alpha=0.1)"; is used to introduce non-linearity to solve the gradient fading problem; such a problem occurs in specially in very complex networks, where during backpropagation, gradients are used to update the network's parameters, but during they get diminished to an extremely small size
4. **Convolutional Layer III - "**MaxPooling2D (pool_size=(2, 2))"; this layer performs and averaging of each feature map, thus reducing spatial dimensions
5. **Flattening Layer – "**Flatten()", which flattens a 2D layer into a 1-dimensional vector.
6. **Fully Connected Layers:** - "Dense(256)", "Dense(128)", "Dense(64)"; these layers are fully connected, helping to learn complex patterns from the flattened output of the flattening layer.
7. **Output Layer** - "Dense(10, activation='softmax')"; this finally layer corresponds to one of the 10 classes from the GTZAN dataset; the "softmax" function is used to assure that

After processing the model is saved under the .keras format and used to develop the recommendation system.

The architecture suggested by the authors of the paper "Music Genre Classification and Recommendation by Using Machine Learning Techniques" [2] is a complex one. After analysis of multiple variations of architecture, a simpler method would benefit the comparative nature of this paper, and thus a third architecture, namely another CNN is proposed by this paper.

iv. Less Complex CNN Architecture

As stated above, the model presented in the paper by A. Elbir [2] is a complex one, and thus simplifications can be made, without compromising the accuracy of the final model. This paper proposes the following, that only the MFCC coefficients (13) are to be extracted and the CNN architecture should be the following:

1. **Dense Layer –** the initial layer is fully connected one with 256 neurons
2. **Hidden Layer (Dense Layer) -** a second dense layer composed of 128 neurons, helping the model learn more complex patterns
3. **Hidden Layer (Dense Layer) -** a third dense layer encompassing 64 neurons, allowing for increasingly abstract patterns
4. **Output Layer -** Dense Layer with the neurons equal to the number of classes defined by the dataset, equaling 10 in this case; besides that, the function softmax is also included in this layer, function described in the representation of the previous CNN structure

The model proposed here should only be taken in as a reference, while it may be simple, the lack of convolutional layers, between the dense layers, leads to the phenomenon known as overfitting, presented in "Section III" of this paper.

V. EXPERIMENTAL RESULTS

This section of the paper is split into the following sections regarding experimental results: the development of the ML model used to classify the sample music and the practical results obtained using them.

i. The Development of the ML Models

For the development of the Models, rigorous testing has been done in order for the models to achieve as high of an accuracy as possible.

The features chosen to be extracted were picked after multiple test runs, as they are most representative of the audio samples, and thus are good candidates for creating a classification model.

1. The Support Vector Machine Model

In the case of the SVM model the code has been run for a total of 40 times (fits). This number can be calculated as such:

$$n\_splits * C(Hyperparameters) = nr\_fits \quad (1)$$

The number of fits, as presented in "Eq. (1)", represents the combination of all the Hyperparameters (4(C)*2(kernel types) = 8) defined in "*Table II*" multiplied by the number of splits in the data. The number of splits, has been arbitrarily determined, after multiple testing to represent 5 folds of the data. This number depicts the way, the cross-validation process will split the data for training and testing, in this case 4 folds of the total samples will be used in the training part of the pipeline while the remaining fold will used to evaluate the model. Because the number of splits was set to 5, it also determines that the code will be run 5 times, each time using another fold for the validation test.

According to the output of the code, the Best Model Hyperparameters tested were the following: *C = 100 & Kernel = Radial Basis Function Kernel*. These parameters where chosen as the best after the grid – search with cross – validation, and it is to be expected, because 100 was the highest tolerance between achieving a low training error and a low testing error. A higher C value would have led to overfitting, but this way, the model accepts some errors in order to avoid this phenomena. The RBF Kernel is also more complex than the linear one, making it more suitable for the

classification application presented by this paper, as it is able to better capture non-linear relationships in the data.

TABLE III. SVM MODEL RESULTS

Overall Data	SVM Model		
	Nr of fits	***Training accuracy***	***Testing accuracy***
	40	0.998	0.7

With this combination of Hyperparameters a *Training Accuracy of 0.998 and a Testing Accuracy of 0.7* have been achieved by the model, as it can be deduced from "*Table III*". The high accuracy for the training of algorithm, might present a small amount of overfitting, but the testing accuracy is high enough to develop a classification application.

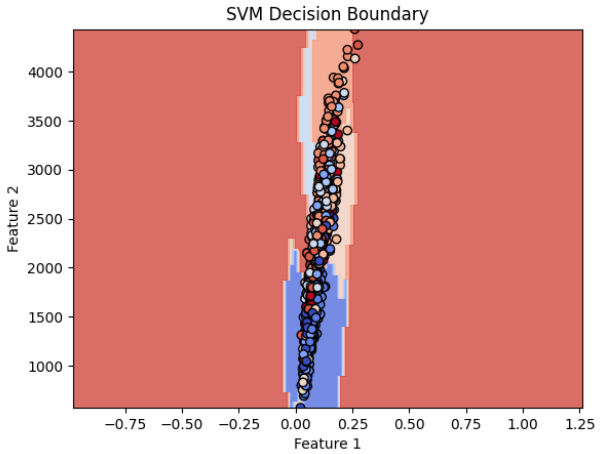

Fig. 3. SVM Decision boundary graph (first 2 features) - All the Samples from GTZAN

2. The Convolution Neural Network Model (CNN)

The CNN model proposed in paper [2] has been arbitrarily chosen to run for 500 times in total. The code was run for a number of 20 epochs, with 25 steps per epoch. An epoch, in the context of neural networks, refers to a complete pass trough all the training subset.

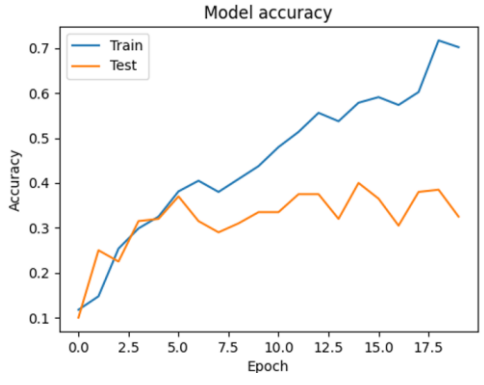

Fig. 4. CNN Model Accuracy Graph over Epochs

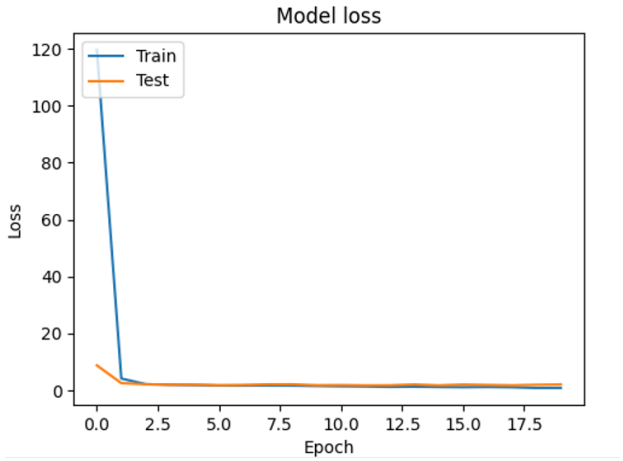

Fig. 5. CNN Model Loss Curve over Epochs

Using the data extracted from Figure 4. And Figure 5, a table has been constructed.

TABLE IV. PROPOSED CNN MODEL RESULTS

Overall Data	Proposed CNN Model		
	Nr of iterations	***Testing loss***	***Testing accuracy***
	500	2.1	0.33

Despite the complex architecture of the model, as presented in "Section IV" of this paper, the testing of the model has presented lacking results. The best result has been obtained during the 20/20 Epoch with an accuracy of 0.71 and a loss coefficient of 0.8, but the final data obtained, according to "*Table IV*", after the model has been trained, states that the testing loss achieved was 2.1 while the accuracy 0.33, which are undesirable values. The high testing loss obtained indicates that, on average, the model's predictions were off by a significant margin, meaning that there is no significant minimization between the predicted outputs and the ones obtained from unseen data, data that the model hasn't used for its training. The low accuracy indicates that this model has a very low accuracy, which should be corrected in order to make it suitable for classification tasks.

The model presented as a reference in this paper, has scored a better overall score, while it might not be so complex, multiple tests have been conducted, so that the complexity of the model could be balanced with the accuracy and loss obtained in the end.

As a first iteration, the CNN had been designed to be even more complex, than the reference one presented in Section 4 of the "*Implementations*" part of this article. Between the dense layers, multiple dropout layers were introduced. Dropout is a regularization method technique commonly used in neural networks to prevent overfitting. With the 50% dropout layers introduce between the dense, hidden layers, each neuron in that layer would have the probability of 0.5 to be dropped out, or deactivated in the current iteration, encouraging the algorithm to form more robust connections, making more generalizable representations. The testing and training accuracy obtained which such a model, while extracting 8 different features (MFCC (Mel-Frequency Cepstral Coefficients), Chroma STFT (Short-Time Fourier

Transform), Spectral Centroid, Spectral Bandwidth, Spectral Rolloff, Spectral Contrast, Spectral Flatness, Root Mean Square Energy) accounted for a merely 0.1 testing and 0.1 training accuracy, which are really bad results, and such a model could never be used in practice.

Several more attempts at simplifying the model, have lead to the creation of the one used as a reference. This model has proved to be simple, but still left room for improvement

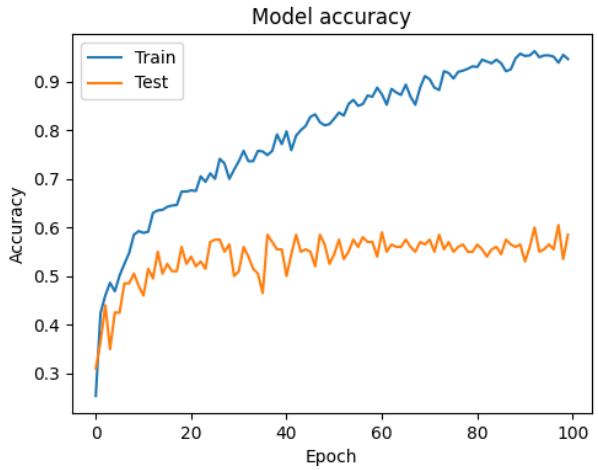

Fig. 6. Simplified CNN Model Accuracy Curve over Epochs

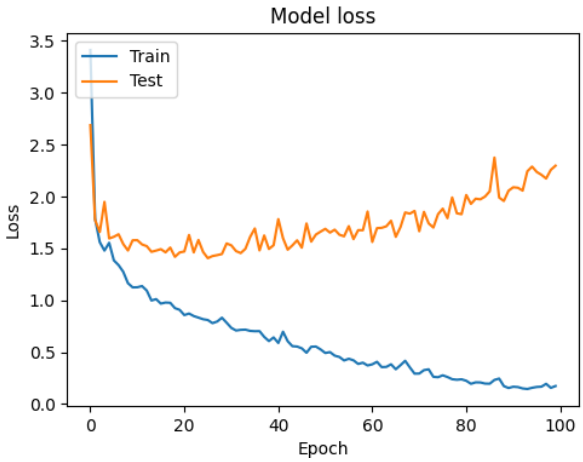

Fig. 7. CNN Model Loss Curve over Epochs

Using the data extracted from Figure 6. And Figure 7, a table has been constructed.

TABLE V. SIMPLIFIED CNN MODEL RESULTS

Overall Data	**Simplified CNN Model**		
	Nr of iterations	***Training accuracy***	***Testing accuracy***
	500	0.998	0.59

"*Table V*" presents the results obtained by training the simplified CNN model for 100 epochs with a batch size of 32, meaning that for each epoch 32 samples from the GTZAN dataset are processed, before updating the weights based on the computed loss.

The high training accuracy of almost 100% in combination with the lower testing accuracy of 59% suggest that the model is performing very well on the tested data while not so great over unseen data, again pointing out some overfitting issues.

VI. CONCLUSION

From the implementations tested, to create a music classification algorithm, a comparative table has been devised.

TABLE VI. COMPARATIVE VIEW OF THE RESULTS

Model	***Training accuracy***	***Testing accuracy***
SVM	*0.998*	***0.7***
Proposed CNN	*0.61*	***0.33***
Simplified CNN	0.998	0.59

From "*Table VI*" it can be easily deducted that the results obtained, after implementing every method, have proven, in this case that the SVM model has performed the best.

It can also be deduced that the algorithms still have room for improvement, as the more simplistic models (SVM, Simplified CNN) suffer greatly from overfitting, which reduces their overall accuracy.

Future improvements are in measure, in order to increase the efficiency of these algorithms. Some strategies would involve the following:

- **Using a larger dataset**, that includes more varying audio samples
- **Data Augmentation,** by adding noise over the audio samples, for example
- **Batch Normalization,** referring to the activation of each layer, making the network more stable
- **Early Stopping,** by stopping the model before the performance starts to drop, due to overfitting
- **More Regularization,** adding more weight decay penalties, encouraging more robust connections between layers

By introducing such layers in the model, the results may find significant better results, although it is important to notice that architectures that are too complex may fail at the same purpose as simpler ones. As presented in the

"*Experimental Results Section*" the model proposed in paper [2], the complexity of the system was much greater than the simpler system described in this paper. Despite the differences between the complex one and the simpler, reference example the final accuracy of the second model presented better possibilities for future improvements.

It is important to note that a balance between complexity and loss must be taken into consideration when creating such a machine learning model, to make it viable for real-world applications.

REFERENCES

[1] L. Moysis et al., "Music Deep Learning: Deep Learning Methods for Music Signal Processing—A Review of the State-of-the-Art," in IEEE Access, vol. 11, pp. 17031-17052, 2023, doi: 10.1109/ACCESS.2023.3244620.

[2] A. Elbir, H. Bilal Çam, M. Emre Iyican, B. Öztürk and N. Aydin, "Music Genre Classification and Recommendation by Using Machine Learning Techniques," 2018 Innovations in Intelligent Systems and Applications Conference (ASYU), Adana, Turkey, 2018, pp. 1-5, doi: 10.1109/ASYU.2018.8554016.

[3] A. Patel, "Ashishpatel26/Tools-to-design-or-visualize-architecture-of-neural-network: Tools to design or visualize architecture of Neural Network," GitHub, https://github.com/ashishpatel26/Tools-to-Design-or-Visualize-Architecture-of-Neural-Network (accessed May 10, 2024).

[4] Andrada, "GTZAN dataset - music genre classification," Kaggle, https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification (accessed May 11, 2024).

[5] V. Alto, "Visualizing SVM with python," Medium, https://medium.com/swlh/visualizing-svm-with-python-4b4b238a7a92 (accessed May 11, 2024).