Musical Genre Classification Using Support Vector Machines, K Nearest Neighbors, and Deep Neural Networks

By Ayush Shenvi Pissurlenkar, Sravya Jevisetty, and Kenechukwu Onyeachonam

Abstract

This project focuses on developing a musical genre classification program that uses machine learning classification models, specifically K Nearest Neighbors, Support Vector Machines, and Deep Neural Networks. This program uses the GTZAN Genre Dataset and 1000 audio files for feature extraction and classification. It contains ten different genres of music, each with 100 audio files of 30 seconds, which we further divided into 3 seconds files. The high accuracy scores from each model demonstrate that they successfully correctly classified each audio file into one of the ten pre-selected musical genres.

Introduction

Music classification is music information retrieval that involves classifying music into labels, such as genre. To classify the music into these labels, one would have to use the data the music provides, such as tempo, beats per minute, etc. This is plausible because studies have discovered that songs under the same musical genre share certain characteristics, such as similar instruments used, similar rhythmic patterns, and similar pitch distributions (Li et al., 2003). A popular application for assigning music to a specific label would be the music application Shazam. Shazam is known for extracting the title and artist of a song from only listening to it for a few seconds. It utilizes the song's peak in amplitude within its spectrogram and other features to match audio with specific identifying information through neural networks (Pelchat & Gelowitz, 2020).

In general, music classification usually involves two steps: feature extraction and multi-class classification. Feature extraction involves retrieving the music signals information that represents the music well, does not require plenty of storage, and does not need much computation. Multi-class classification requires an algorithm that can identify the labels of each music signal with respect to the extracted features (Li et al., 2003).

Problem Statement

The goal of our project was to create a program that can classify different audio files into one of the musical genres out of the ten provided in the dataset, which are the following: blues, classical, country, disco, hip hop, jazz, metal, pop, reggae, and rock. To achieve this, we utilized three machine learning classification algorithms, specifically K Nearest Neighbors, Support Vector Machines, and Deep Neural Network, within the Python programming languages.

Project Relevance

Our project focuses on classifying songs under a musical genre from a list of previously-selected genres. This topic would resolve the issue of pieces being classified incorrectly or being placed under multiple genres. For instance, Spotify, one of the most popular music streaming platforms, has over 5,000 distinct genres (Wahab, 2022), of which each of its 80 million tracks (to date) are part of (Spotify, 2022). It is typical for individual tracks to be labeled under more than one genre. But having over 5,000 to choose from can make this type of music classification difficult, especially as more songs get added to a streaming platform. For example, a song can be classified under too many genres due to an artist taking inspiration from other genres. As a result, it can create a problem for the artists by making it harder to anticipate the type of sound their listeners would expect or like to listen to from them. Narrowing the genres down into a couple or even one can help fix this problem.

Additionally, the number of songs that are available for the public to stream is rapidly growing, so it is essential to correctly categorize music for "organization, search, and retrieval" (Vishnupriya & Meenakshi, 2018). This would allow streaming services and other music-sharing platforms to provide an organized dataset of available music, which would make it easier for users to find their favorite songs and create playlists. Also, it would allow artists to accurately advertise themselves under a certain genre to attract more listeners interested in that type of sound.

Dataset and Features

This project utilized the GTZAN Genre Dataset (Olteanu, 2020). It is one of the most used public datasets for Music Genre Recognition (MGR). It contains files collected from 2000-2001, from multiple different sources such as CDs, radio, microphone recordings, etc. This data contains a genre folder containing 10 unique genres with around 100 audio files, each with a duration of 30 seconds.

In order to use these audio files, we needed to process them and extract their features. We utilized the python module, Librosa, in order to analyze audio signals. We extracted 52 features in order to create the 2D dataset we used to train and test our model. We looped through every audio file and utilized Librosa to get the Zero crossing rate, Spectral Centroid Mean and Variance, Spectral Rolloff Mean and Variance, Spectral Bandwidth Mean and Variance, Chromagram Mean and Variance, and each of the 20 Mel-Frequency Cepstral Coefficients. Each one of these features allows us to understand different parts of the music file and we used various Librosa methods in order to extract each one. The zero crossing rate is the rate that the signal changes sign, from negative to positive or reverse. The spectral centroid tells us where the center of mass for the sound is located. In order to calculate this, the mean and the variance of the frequencies are found. The spectral roll-off is the shape of the signal. For each frame in the signal, we calculate the frequency below an imputed percentage of the whole spectral energy and then get the mean and variance of all of the frames. Next, we got the Mel Frequency Cepstral Coefficients, which represent the entire shape of the spectral envelope, essentially modeling the human voice characteristics. Finally, we got the chroma frequencies which represent the audio music file in 12 distinct semitones. We also divided up each 30 second audio file into 3 second files to further broaden our dataset, thus allowing the models to predict a genre given only 3 seconds of a song.

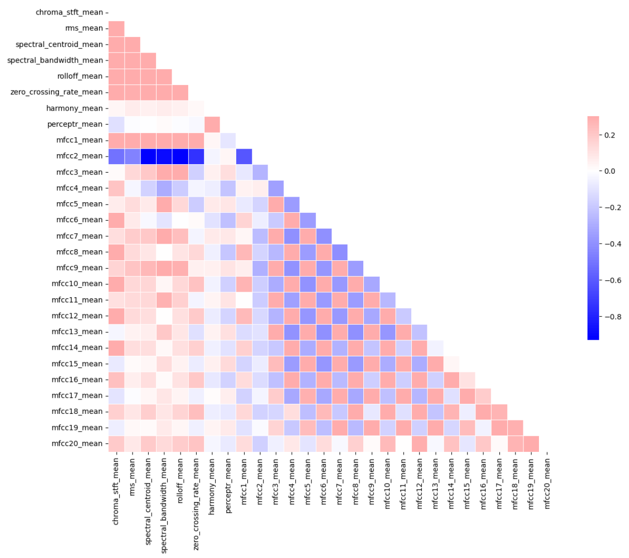


Figure 1.1 - This image shows a correlation matrix graph between all the pairs of extracted features .

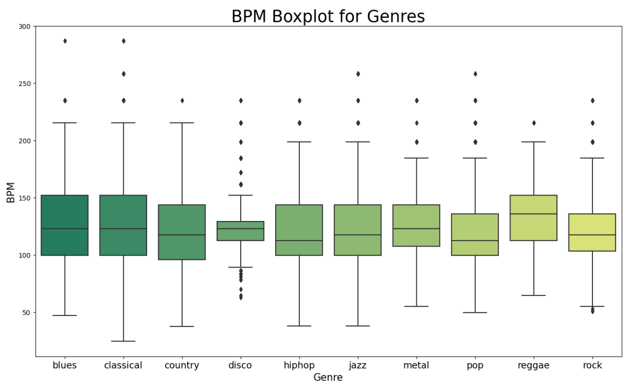


Figure 1.2 - This boxplot graph displays the median, maximum, and minimum values for the beats per minute feature for the group of audio files under each musical genre.

Technical Approach

In the following sections we will go in-depth into the 3 proposed models (KNN, SVM, and DNN) developed to produce various results.

Model 1: K Nearest Neighbors (kNN)

K Nearest Neighbors is a common machine learning algorithm that can label a sample under a specific class by analyzing the k training samples nearest in distance to the sample. The class with the majority vote of its k-nearest neighbors “wins” and is returned as the predicted class.

In terms of our project, when a song snippet is fed into our KNN, based on the most common genre of the k nearest neighbors around the inputted song snippets feature vector, that genre is returned. In order to implement this model we decided to utilize the KNeighborClassifier() method from the Sklearn python library. Initially, we set up our KNN and fit it to 80% of our data, also known as our training data. Then we proceeded to standarize our X_train and X_test data using the StandardScaler() which helps remove the mean and scaling to unit variance for the features. Which means it rescales the distribution of the values within our dataset so that the mean of the values is 0 while the standard deviation is 1.

In order to test how well our initial KNN worked we utilized the score function inside of the KNeighborsClassifier() class. When testing our initial KNN on both our training data the Train score was 0.9365 and the Test data score was 0.8777. While this was a good accuracy we still wanted to see if we could improve it.

To improve this initial model, we utilized hyperparameter tuning and cross-validation.

Hyperparameter tuning allows us to adjust the parameters in the KNN model to find the best combination of hyperparameters to result in the most accurate prediction. To implement the hyperparameter tuning, we utilised the built-in function that does a grid search among the best combination of input hyperparameters.

Parameters for Grid Search

Parameter	Options
n_neighbors	3, 5, 10, 15, 20, 25
weights	'uniform', 'distance'
metric	'euclidean', 'manhattan'
algorithm	'auto', 'ball_tree', 'kd_tree', 'brute'

We utilized cross-validation in order to get the most correct accuracy score. In order to protect against overfitting in the initial prediction model the model is tested on a variety of different training and testing data subsets.
In the end the grid search returned the following parameters as the best combination for KNN.

{'algorithm': 'auto', 'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}

Creating a model with the parameters resulted in a Train score of 0.999 and a Test score of 0.92, which is a significant improvement from the previous model.

Model 2: Support Vector Machines (SVM)

The Support Vector Machine model is a popular supervised machine learning model for classification (or regression) analysis. This model functions by mapping data into a high-dimensional feature space to categorise the data points. It creates a decision boundary using the input dataset to maximise the minimal distance from data points to the decision boundary. In our SVM model, we utilise linear, radial basis function and polynomial kernel to find out which kernel works best with our dataset.

We normalize the X data from the dataset using a MinMaxScaler(), which transforms all features by scaling them to within the default range of (0, 1). This helps reduce the distance between observations and helps maximize the distance between the closest data points to create a decision boundary for the model. Then we used the data to train an SVM model random, with its kernel as ‘rbf’ with C=10 and gamma=0.1. This resulted in us having an accuracy of 0.78. More information in **Appendix 1,** which contains the classification report of the SVM model. We utilized hyperparameter tuning and cross-validation to improve this initial model, as we did with KNN.

Parameters for Grid Search

Parameter	Options
C	0.1, 1, 10, 100, 1000
gamma	1, 0.1, 0.01, 0.001, 0.0001
kernel	'rbf', 'poly', 'linear'

In the end the grid search returned the following parameters as the best combination for SVM.
{'C': 1000, 'gamma': 1, 'kernel': 'rbf'}

Creating a model with the parameters resulted in an accuracy of 0.93 which is a significant improvement from the previous model. The grid search results can be seen in **Appendix 2** & the classification report in **Appendix 3**. This can also be seen through the confusion matrix shown below, where most of the data is on the diagonals showing most of the data is correctly predicted.

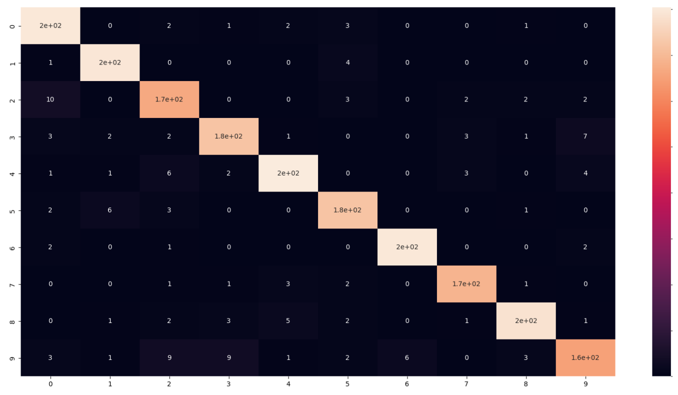


Figure 2 - This heatmap shows the confusion matrix created by the grid search SVM model.

Model 3: Deep Neural Network (DNN)

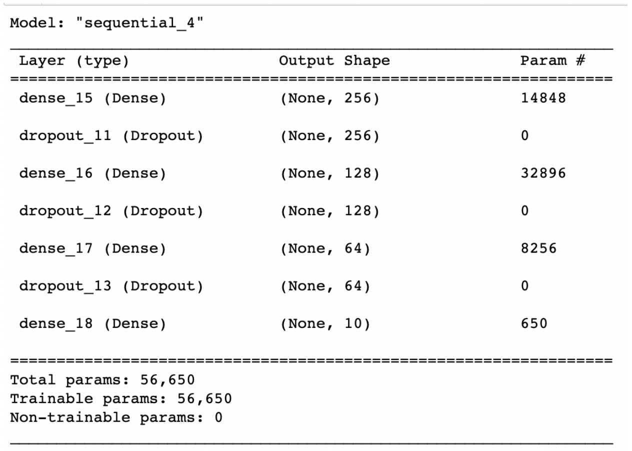
A deep neural network (DNN) is a type of artificial neural network (ANN) that is composed of multiple layers of interconnected artificial neurons, or "nodes." ANNs are inspired by the structure and function of the human brain and are designed to recognize patterns and make decisions based on input data.

A DNN typically consists of an input layer, one or more hidden layers, and an output layer. The input layer receives input data, which is then processed by the hidden layers using weights and biases to produce output data. The output data is then passed through the output layer, which produces the final output of the network.

For our DNN we will be setting loss as sparse categorical cross-entropy, optimizer as adam and experimenting with the rest of the parameters. For this model we will be setter all Dense layers to relu and having the activation function as softmax. A Dropout layer is also present after each normal layer to lower the time taken to train the model.

To prepare our data we first use label encoding to change the target labels to integer types to make it easier for the model to train. Then we used StandardScaler() to standarize the data similar to the KNN model.

To create a baseline for DNN we randomly created and tested the model in Figure 3, with a batch_size=16, and epochs=100.



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 256)               14848

 dropout_11 (Dropout)        (None, 256)               0

 dense_16 (Dense)            (None, 128)               32896

 dropout_12 (Dropout)        (None, 128)               0

 dense_17 (Dense)            (None, 64)                8256

 dropout_13 (Dropout)        (None, 64)                0

 dense_18 (Dense)            (None, 10)                650

=================================================================
Total params: 56,650
Trainable params: 56,650
Non-trainable params: 0
_________________________________________________________________
```

Figure 3 - Sequential Model of the randomly created DNN.

This resulted in an average test loss of 0.49 and an accuracy score of 0.919. A plot with all losses can be seen in **Appendix 4**.

To improve on this model we once again used Hyperparameter tuning with grid search and cross-validation, the parameters are shown below.

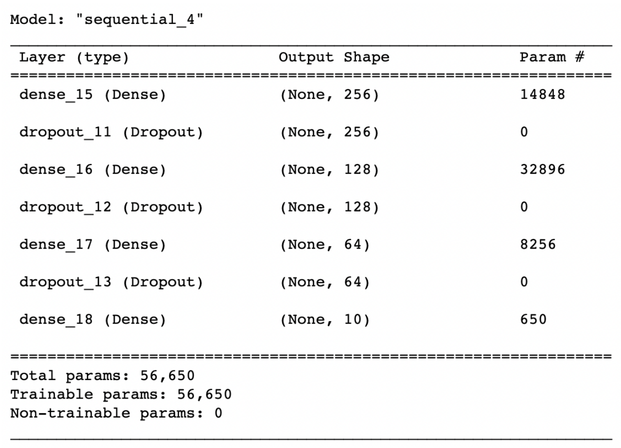
Parameters for Grid Search

Parameter	Options
batch_size	16, 32, 64
epochs	10, 100, 1000
dropout_rate	0.0, 0.10, 0.20
hidden_layers	1, 2
neuron_array	[256, 128, 64], [128, 64]

From grid search the following parameters were returned.

{'batch_size': 64, 'dropout_rate': 0.2, 'epochs': 1000, 'hidden_layers': 2, 'neuron_array': [256, 128, 64]}

Which we then used to produce a new DNN model:



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 256)               14848

 dropout_11 (Dropout)        (None, 256)               0

 dense_16 (Dense)            (None, 128)               32896

 dropout_12 (Dropout)        (None, 128)               0

 dense_17 (Dense)            (None, 64)                8256

 dropout_13 (Dropout)        (None, 64)                0

 dense_18 (Dense)            (None, 10)                650

=================================================================
Total params: 56,650
Trainable params: 56,650
Non-trainable params: 0
_________________________________________________________________
```

Figure 4 - Sequential Model of from the grid search parameters.

This model resulted in a test loss of 0.47 and an accuracy score of 0.932 which while not a large increase is still a better score.

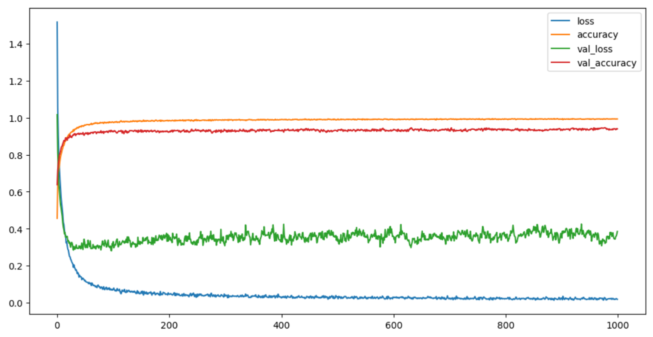


Figure 5 - Plotted data from the fit of the DNN model.

Results and Conclusion

From our EDA we observed that there was great correlation between the mfcc2_mean and most of the other features, which leads us to believe that it holds the highest feature importance in our dataset.

From our data, we understood that we needed to select models that show a good relationship with observed data points. Thus, we chose KNN models which classify data with respect to observation around a selected data point which leads it to be a great classifier for musical data. This is because we see great similarities between the music of the same genres depend in accordance to features such as tempo, precision, instruments, harmonics, etc.

This is also a reason why we chose the SVM model which closely looks at observations and creates a decision boundary that maximizes the minimum distance between the plane and the closed observed data points. SVMs are effective in high-dimensional spaces, thus music tracks can have a great number of features and SVMs are able to handle large feature spaces effectively. It is also really robust to overfitting and can handle imbalanced datasets.

DNNs are able to learn complex patterns in data due to their large number of layers and the ability to adjust the weights and biases of the connections between the neurons. This makes them well-suited for tasks where the relationships between the features and the labels are complex. DNNs can learn directly from raw data, such as audio or image data. This is useful in music genre classification, where the raw audio data can be used as input to the network, rather than extracting a set of features from the data. Overall, DNNs are a powerful tool for music genre classification due to their ability to learn complex patterns, handle large datasets, and learn directly from raw data.

Now comparing the scores that we produced using our models and grid search algorithms.

Accuracy Scores

Model	Accuracy
KNN	0.921
SVM	0.928
DNN	0.932

As we can see from the results the DNN model was able to produce the best accuracy score, this is due to the fact that DNNs are able to learn complex patterns in data due to their large number of layers and the ability to adjust the weights and biases of the connections between the neurons. This makes them well-suited for tasks where the relationships between the features and the labels are complex. In contrast, SVMs and k-NN are limited in their ability to learn complex patterns. DNN are also able to handle and process large datasets efficiently and can learn from a large number of examples. SVMs and k-NN may struggle with large datasets due to the computational complexity of the algorithms. DNNs can also learn directly from raw data, such as audio or image data. This is useful in music genre classification, where the raw audio data can be used as input to the network, rather than extracting a set of features from the data. SVMs and k-NN, on the other hand, typically require the input data to be transformed into a set of features before they can be used for classification.

We developed three models to classify a song's audio snippet's genre. We extracted a multitude of features from approximately 1000 songs, split the data up into testing and training subsets, and normalized the feature data. We then developed a k Nearest Neighbors model, a Support Vector Machine model, and finally a Deep Neural Network model. After developing each initial model, we went back and utilized the results of running a grid search on each model along with cross-validation to ensure the most optimal accuracy for each model. After analyzing each model, we found that the DNN model has the best accuracy in classifying the musical genre for each audio file from the dataset. Due to these reasons, as we mentioned before.

Future Recommendations

For further work on this topic I believe that we would need to create a more in-depth grid search algorithm, as there are so many more parameters that can change the way the model behaves. At the current point where we have models with extremely high accuracy, we can only improve our models using extremely precise hyperparameter tuning. We also believe that having a larger dataset, and one with more recent songs will allow our models to have a greater meaning in a real-life situation.

We also believe that we should have focused more on giving more weight to features that have high importance than others.

Thus, the improvements we would like to in the future would be:

- Create a more advanced grid search algorithm that can tune more parameters. Allowing us to change our models with extreme precision.
- Adding more features, or even refining our features to show more importance/give more weight to features that seem to have a higher correlation to our data could help elevate our model.
- Model ensembling: combining classifiers by voting or averaging to improve performance

References

About Spotify. Spotify. (2022, October 25). Retrieved October 28, 2022, from https://newsroom.spotify.com/company-info/

Li, T., Ogihara, M., & Li, Q. (2003). A comparative study on content-based music genre classification. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '03*. https://doi.org/10.1145/860435.860487

How SVM works. (2021, August 17). Retrieved December 15, 2022, from https://www.ibm.com/docs/en/spss-modeler/saas?topic=models-how-svm-works

Pelchat, N., & Gelowitz, C. M. (2020). Neural network music genre classification. *Canadian Journal of Electrical and Computer Engineering*, *43*(3), 170–173. https://doi.org/10.1109/cjece.2020.2970144

Sklearn.neighbors.kneighborsclassifier. scikit. (n.d.). Retrieved December 15, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Tzanetakis, G., & Cook, P. (2002). Musical genre classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, *10*(5), 293–302. https://doi.org/10.1109/tsa.2002.800560

Vishnupriya, S., & Meenakshi, K. (2018). Automatic music genre classification using Convolution Neural Network. *2018 International Conference on Computer Communication and Informatics (ICCCI)*. https://doi.org/10.1109/iccci.2018.8441340

Wahab, Y. (2022, January 12). *9 music genres on Spotify you might not have heard of before*. Bandwagon. Retrieved October 28, 2022, from https://www.bandwagon.asia/articles/9-eclectic-spotify-genres-you-may-not-have-heard-of-before-wrapped-every-noise-at-once-lowercase-stomp-and-holler-pixie-skwee-catstep-ninja-Solipsynthm-wonky-escape-room-2021

Olteanu, Andrada. "GTZAN Dataset ." *Kaggle*, 24 Mar. 2020, https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification.

Won, M., Spijkervet, J., & Choi, K. (2021). *Music Classification: Beyond Supervised Learning, Towards Real-World Applications*. https://doi.org/https://doi.org/10.48550/arXiv.2111.11636

APPENDIX

```
              precision    recall  f1-score   support

       blues       0.75      0.79      0.77       208
   classical       0.87      0.97      0.92       203
     country       0.67      0.74      0.70       186
       disco       0.65      0.70      0.67       199
      hiphop       0.86      0.76      0.81       218
        jazz       0.83      0.82      0.83       192
       metal       0.84      0.91      0.87       204
         pop       0.85      0.87      0.86       180
      reggae       0.81      0.71      0.75       211
        rock       0.68      0.56      0.61       197

    accuracy                           0.78      1998
   macro avg       0.78      0.78      0.78      1998
weighted avg       0.78      0.78      0.78      1998
```

Appendix 1 – Classification report of the initial SVM model

```
mean_fit_time                                         0.747037
std_fit_time                                           0.00346
mean_score_time                                       0.295626
std_score_time                                        0.013628
param_C                                                   1000
param_gamma                                                  1
param_kernel                                               rbf
params               {'C': 1000, 'gamma': 1, 'kernel': 'rbf'}
split0_test_score                                      0.91375
split1_test_score                                       0.9125
split2_test_score                                     0.922403
split3_test_score                                     0.913642
split4_test_score                                     0.907384
split5_test_score                                      0.90363
split6_test_score                                     0.909887
split7_test_score                                     0.923655
split8_test_score                                     0.913642
split9_test_score                                     0.914894
mean_test_score                                       0.913539
std_test_score                                        0.005773
rank_test_score                                              1
Name: 60, dtype: object
```

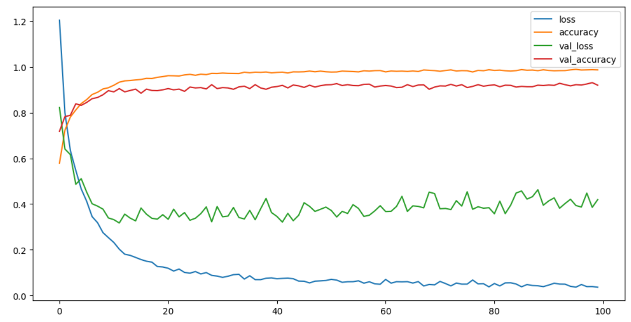
Appendix 2 – Grid Search index of the best SVM model

```
              precision    recall  f1-score   support

       blues       0.90      0.96      0.93       208
   classical       0.95      0.98      0.96       203
     country       0.87      0.90      0.88       186
       disco       0.92      0.90      0.91       199
      hiphop       0.94      0.92      0.93       218
        jazz       0.92      0.94      0.93       192
       metal       0.97      0.98      0.97       204
         pop       0.95      0.96      0.95       180
      reggae       0.96      0.93      0.94       211
        rock       0.91      0.83      0.87       197

    accuracy                           0.93      1998
   macro avg       0.93      0.93      0.93      1998
weighted avg       0.93      0.93      0.93      1998
```

Appendix 3 – Classification Report of the best SVM model



Appendix 4 – Plot for the initial DNN model