Machine Learning - Short Project

Théo Desquins - Stéphane Maillot

May 2017

Contents

Motivations

Music type classification was first used by music re tailor (to guide customers through the store), by radios and television and is today a necessity due to the digitizing of music and the raise of internet musical platforms like Deezer, Spotify, etc... which has to manage and suggest a big quantity of song. [1, 6]
Music genre isn't a fixed semantic science. Every individual has its own perception of the music even if we can distinguish some patterns that lead to the building of music type. From those similitude, an individual may like other musics of the same type.

Objectives

Our objective isn't to achieve performances comparable to the MSN Music Search Engine (MMSE) (2001) (we neither have the money nor the data for this) but to extract relevant features and to compare the result of different classification methods on them.
The features are extracted from a dataset of music from MARSYAS prepared for supervised learning and containing (10 different genres). Several papers used this dataset to test music type classifier and restricted the study to 4 or 5 type of music due to poor result while trying to classify all the set. Our aim is to compare several classification methods on the 10 classes problem.

State of the art

Classifiers

The number of categories and sub categories is not fixed and more than one taxonomy for music classification are proposed.

- AllMusicGuide : 531 genres
- Amazon : 719 genres
- MP3 : 430 genres

Two type of classification exists, manual classification, performed by a musicologists, and automatic classification. One work of manual classification has been performed by MSN music to develop the MMSE in 2001 [2] which compare manual and automatic genre classification and which found that the accuracy of the manual classification vary between 92% and 95% while automatic classification between 88% and 94% .

A lot of papers are dealing with this genre classification problem. In *Factors in automatic musical genre classification of audio signal* [4] which analyzes the accuracy of genre classification for SVM, Multi-category Proximal Support Vector Machine (MPSVM) or LDA and using the first 5 Mel-Frequency Cepstral Coefficients (MFCC) and 4 features computed from the Short Time Fourier Transform (STFT). The result of this study generated with 100 music samples for each of the 10 genres, 90 % of data used for training and 10 % for test, results in an accuracy between 20% and 70% depending the combination of features.

Another study was made at Standford and focus on one MFCC feature and try to classify four music genres (classical, jazz, metal, and pop) using k-nearest-neighbour, k-means and multiclass SVM with 100 music samples for each genre. 7O% of the data is used for training and 30% for testing.
The best accuracy is obtained using the neural network. All these works use MARSYAS as feature extraction software.

Features

In this project we will need to extract relevant features. From the previously mentioned papers [7], some features need to be explained.

- Most of the features use STFT which is a mathematical tool to study the evolution of the frequency power spectrum over time.
- Energy: mean of the root mean square energy of the signal. It takes into account the human perception of the squared amplitude of sound waves.

$$\sqrt{\frac{\int_t x^2}{\Delta t}}$$

- MFCCs are coefficients computed from the short-term power spectrum of a sound.[5, 1]

Proposal

Our proposal is then to extract as much relevant features as possible. These features will be based on music theory knowledge in order to be as accurate as possible in this genre classification problem.
We will then use these features in different types of classifiers to compare results we obtain on the multiclass problem.
The classifier we will compare are:

- k nearest neighbor
- single multiclass classifier
- Tree of classifier
- Deep learning

Feature extraction

Database

Before extracting any feature, we need to choose the database to extract features from. We first wanted to use our own music library to create this classifier. The size of this kind of

dataset (few hundreds songs) isn't huge but appears to be enough for what we want to achieve.
However we need a varied and well classified dataset to perform supervised learning so a personal music library doesn't appear to be the best choice. Moreover, as we previously mentioned, mp3 format can deal with a lot of genres and we can't deal with that much different classes.

For these reasons, we searched for a reasonable size database (to stay close from what we could obtain using a well classified personal song library) that is well classified in a limited number of genres. We finally chose the GTZAN genre collection dataset from MARSYAS. This dataset was made by the author of some papers we quoted previously.
This set contains 1000 tracks splited in 10 genres and already cuted into 30sec extracts.

Audio python library

We decide to code this project in python so our first task is to find an audio library that could help us to extract audio features from our songs. After testing some libraries, we chose to only use librosa, a library that allows to extract many features from music tracks.
We especially use the functions to split audio signals into their percussive and harmonic parts. our features will be extracted from them.

Tempo

The most simple feature we can extract is the tempo of the song. Mathematically it corresponds to the dominant frequency of the percussive part. It could be computed as the peak in the Fourier transform of the signal auto-correlation but here we use librosa's tempo function.

Energy

As mentioned previously, we can compute the energy of a signal from its amplitude. Librosa also allows us to compute the energy of an audio signal more easily. We have found that computing the energy ration between percussive and harmonic parts is the most relevant feature to extract from these energies.

Principal pitch

Using librosa, an other interesting functionality is to get the frequency power histogram stacked by pitch (A, A#, B, C...). We then use this histogram to get the principal pitch of the song.
We also use this histogram to compute the succession of notes in the harmonic part of the histogram and there duration. This leads use to a single list of notes that we filter in order to only keep the melody and not the noise composed by very short notes.
So at this point we have an array of notes giving the pitch and duration converted steps according to the tempo (1 step = 1min / tempo).

Principal intervals

In music theory, most songs have a principal pitch and chords. Taking this pitch as reference, we can then compute the different interval switch (from one note to the next one). We search for the 2 principal intervals and keep them as a feature considering that they give information about the scale of the song.

Main interval switching

Similarly we search for interval switching. Indeed, in music the same interval is often used several times (arpeggio) so switching often from one arpeggio to an other should be musically relevant to distinguish between songs.

Rhythmic and melodic graph

As we read in the Merkelbach paper [7], we implemented an algorithm to extract graph from the notes array. It basically consists in counting how many time the song melody switch from A to B (melodic graph) or from a 1/4 step to a 1/8 step. The results of these computations are stored in a 32x32 (1/8th step to 4 steps) matrix for the rhythm and a 12x12 matrix for pitches.

Dimentionnal reduction

This feature extraction process gives us a vector of dimension 1200, as we only have 1000 samples we can't do machine learning on a 1200 dimensional space so we need to reduce the dimension of the feature space.
For that, we have tried to use the PCA and LDA of these features.

PCA

The PCA of these features (figure 1) shows that classes are really hard to separate. For example only the first component seems to allows to distinguish sligthly between classical and other genres.

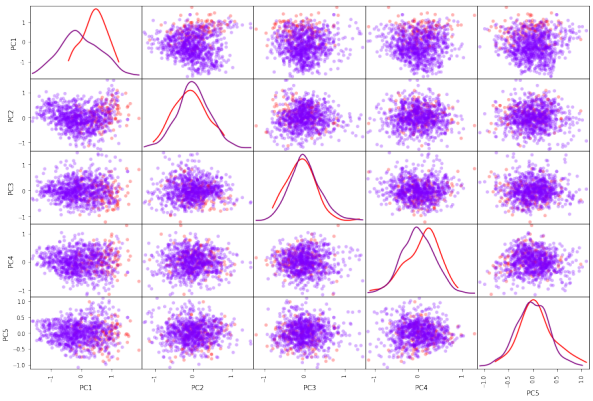

Figure 1: 5 components PCA - classical against all

LDA

The LDA (figure 2) provides features that seems to separate more the classes. However LDA components appear to be the same (or linear combination of them) and only 4 of them seems to be independent from others(excluding the first one).

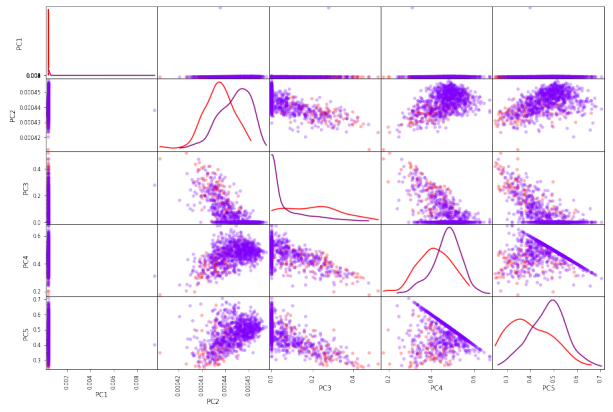

Figure 2: 5 components LDA - classical against all

Preparation of the data

The data are organized as follows for each branch:

- we replace in each class the name of the music genre by a label between 0 and 9
- in each music genre set of samples is randomly internally shuffled
- we pick the sets of music genres we want to use in order to train and test the classifier of the branch.
- 25 % of the samples of each music genre goes to the training set while 75 % goes to the testing set.
- the training set and the testing set are separately randomly shuffled
- we generate a list of label vectors. Each label vector represent a music genre versus all the others selected.

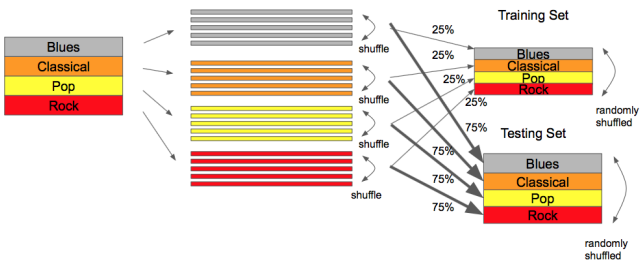

Figure 3: Preparation of the data before the training

K nearest neighbor

Our first attempt to classify the song database is a K-Nearest Neighbors (KNN). We found using 13 nearest neighbor give the best results.
However our results are not satisfying neither with PCA or LDA features. Here are the result for classical against all.

With PCA (figure 4) features the classifier we obtain classify really few sample as positive resulting in only 50% of true classical samples well classified.

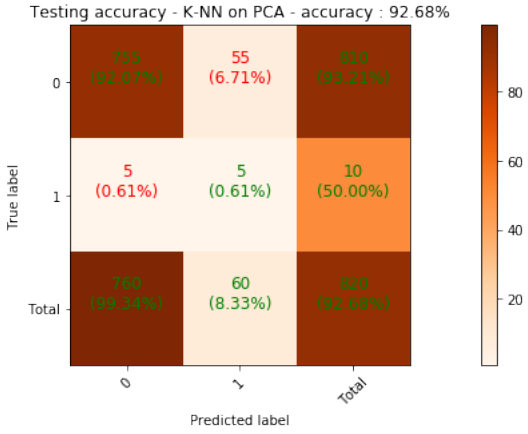

Figure 4: Result of KNN classifier on PCA features for classical against all

With LDA (figure 5) features a lot more classical samples are recognized (70%) but this leads to a high number of false positive.

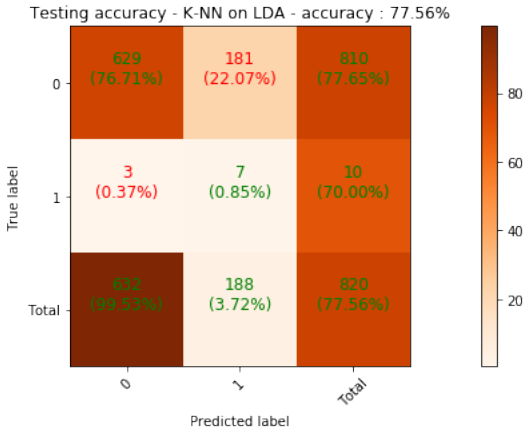

Figure 5: Result of KNN classifier on LDA features for classical against all

Single neural network

A multiple class neural network has been realized with tensorflow (the google library for neural network) to see the global performances of a neural network given the 10 musical genres. We test classifiers with three layers containing the same

number of neurons by increasing number of neurons. For each of the classifiers, ten tests have been realized.

Layer 1	Layer 2	Layer 3	Acc
20	20	20	31,97 %
30	30	30	33,08 %
40	40	40	34,22 %
50	50	50	33,52 %
60	60	60	34,05 %
70	70	70	34,37 %
80	80	80	33,42 %
90	90	90	33,00 %
100	100	100	33,82 %
110	110	110	34,42 %
120	120	120	34,07 %
130	130	130	35,18 %
140	140	140	33,57 %
150	150	150	33,80 %
160	160	160	34,15 %
170	170	170	34,43 %

Table 1: Mean of the accuracy of the unique classifier of the 10 music genres (mean realized over 10 tests for each classifier)

Layer 1	Layer 2	Layer 3	Acc variance
20	20	20	$1.239.10^{-2}$
50	50	50	$1.535.10^{-2}$
70	70	70	1.95610^{-2}
90	90	90	$1.229.10^{-2}$
110	110	110	$1.378.10^{-2}$
130	130	130	$1.263.10^{-2}$
150	150	150	$1.327.10^{-2}$

Table 2: Variance of the accuracy of the unique classifier of the 10 music genres, (variance realized over 10 tests for each classifier)

We can see that the accuracy doesn't exceed 35 % with a variance in the neighborhood of 1.3 %. The value of the accuracy will be compared with the accuracy of the tree of classifier.

Tree of classifiers

The tree of classifier is composed of ten branches. At each one of its nodes a classifier realized with tensorflow distinguish one type of music versus all the music types that haven't its own classifier in one of the previous node.

To find the different elements of the classifier, we have proceeded as follows:

- first, we rise the number of layers for a fixed number of nodes to estimate how many layers we need in order to classify the musics (we obtain almost not one true positive with the different classes tested for two layers, for more layers than 3 we obtain slightly worse result than for 3 layers, so our classifiers have 3 layers)

For one branch of the tree:

- second, for the same number of nodes (100,100,100), we try every classes available for this branch then pick the one that has the best result in term of accuracy with a preference for a minimum of false positive (the false positives can't be classified by the other branches of the tree). For each type of music, 3 trial are done.

- For the class we picked we rise the number of nodes to see approximately where the best performances of the system are attained by rising 10 by 10 the number of nodes in the layer. We obtain a classifier of type [n, n, n]. For each n, 5 trial are done.

- We modify the number of node for each layers in an area around n beginning by the last layer, picking the best performance given by the modifications of the last layer, then we modify the second layer with picked number of node for the third layer, we take the best performance for the second layer, and then do the same for the first layer, taking the nodes obtained in the two previous steps. All the modification are 10 by 10 and 10 test are realized for each combination.

- We go to the next branch

We obtain the following result for each branch from the first to the last :

Classifier	Layer 1	Layer 2	Layer 3
Hip-Hop	100	130	100
Jazz	80	120	130
Classical	60	70	50
Country	120	110	110
Metal	170	140	110
Pop	105	115	110
Blues	57	53	47
Disco	70	70	60
Reggae	40	50	60
Rock	50	100	100

Table 3: Number of neuron at each layer for each classifier, the list of classifier is organized by their order of appearance in the tree

Which give the following result (mean realized over 10 tests under the same conditions , shuffle of the data then training then testing).

We evaluate the performances of the classifiers with the accuracy,the proportion of positive well classified, the proportion of negative well classified. The accuracy is not relevant at the beginning of the tree (for example the first classifier has 750 samples , 75 labeled as 1 and 675 labeled as 0, if all the testing samples are classified as 0 then the accuracy is equal to 90 %). At the beginning (3 first classes) a mean of 20 samples labeled as 0 isn't well classified and for the last classifier, a mean of 35 samples labeled as 0 are miss-classified. We tried to know if this error of the zero labeled was due to the miss classification of all the sample of one class and tried to test the

Class	ACC	PPWC	PNWC
Hip-Hop	88.6 %	60.4 %	97.25 %
Jazz	90.3 %	38.8 %	96.5 %
Classical	92.6 %	61.3 %	96.5 %
Country	85 %	33.2 %	91.88 %
Metal	80.8 %	27 %	92.9 %
Pop	76.8 %	23.73 %	86.66 %
Blues	69.1 %	31.4 %	81.8 %
Disco	63 %	41.86 %	71.2 %
Reggae	70.4 %	58.8 %	69.46 %

Table 4: Performance of each classifier (accuracy, proportion of positive well classified, proportion of negative well classified

classifier in the same conditions, but subtracting one of the genre versus which the class of the branch was classified. The result is that negative miss-classified samples equally come from all the other musical genre. From the fourth branch the performances are lower than the first branches.

The tree is realized as follows:

- The classifiers are trained with the same method than the training of the individual classifier
- The testing is realized as follow, the 750 testing samples goes through the first branch, the samples classified as 1 aren't touched, the samples classified as 0 goes through the second branch and this operation is repeated for each branch till the last one which classify rock versus all musical genre.
- For each branch we take the local proportion of true positive, true negative, false positive, false negative (for the remaining number of sample tested) then the true proportion of true positive, true negative, false positive, false negative (with respect to the global testing population (750 samples))

Then we obtain the result for the final tree of classifier:

Test	Acc
1	31.73 %
2	28.27 %
3	28 %
4	29.60 %
5	30.53 %

Table 5: Accuracy of the tree of classifier given by five tests

From those experiments, we can observe that few of the musical genre can be distinguished from others (in one versus all) using the features we extracted (only Hip-hop, Jazz and Classical). The weakness of the tree method is highlighted by the final experiments. By comparing the evolution of the remaining population of testing samples classified in each branch, we can see a small variation of the population tested for the first three classifier during the test of the tree and a strong variance in the population tested for the classifiers Metal, Country, Blues, which means that the classifiers for those three music genre aren't well chosen or reveal that the tree isn't adapted (indeed the accuracy and proportion of positive well classified is low for those three genre which can signify that from this branch, the genres need a treatment one vs one and not one vs all in order to raise their respective accuracy).

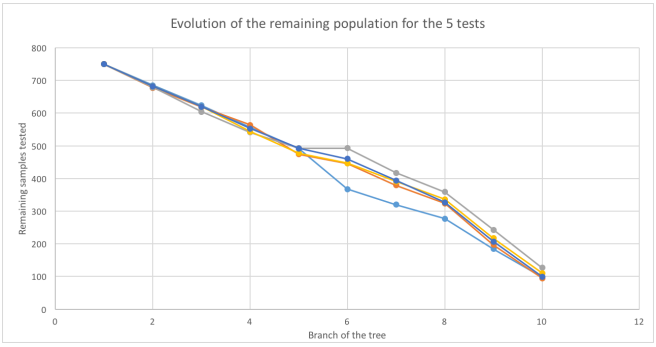

Figure 6: Evolution of the number of samples tested through the branches

Deep learning

As none of our classifiers reached a good accuracy, we read more articles on this subject and it appears that people that claims to obtain good results on genre classification are all using Deep Convolutional Neural Network (DCNN). We found an article [3] where the author explains the structure of the network he uses.
We then managed to create a similar graph on tensorflow to try on our dataset. The structure of the network is the following.

Prepare data

The first part consists in preparing data for learning because this Convolutional Neural Network. The input of this network is a piece of histogram extracted from the STFT of the song. As in the article we extracted 128x128px images from this histogram, all frequencies are kept and down sampled to 128 and the histogram is also down sampled in time (50 samples per second) so we are selecting 2.5sec pieces of the song each time we extract a piece of histogram 7.

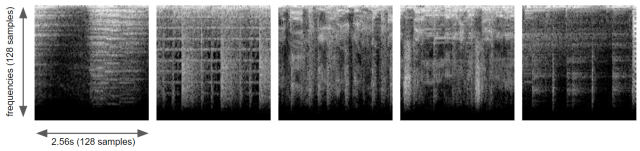

Figure 7: Example of 128x128px histograms we extract from the database

As we need a lot of data we extract 10 histogram for each track, resulting in a database of 10.000 histograms.

Network

We basically (tried to) kept the same structure as in the article. So our tensor flow graph (figure 8) is composed by 4 convolutional layers using 2x2 filters with a stride of 2. Then 2 fully connected layers (with a drop rate of 50% to avoid overfiting) are used to get the final answer to the multiclass problem.

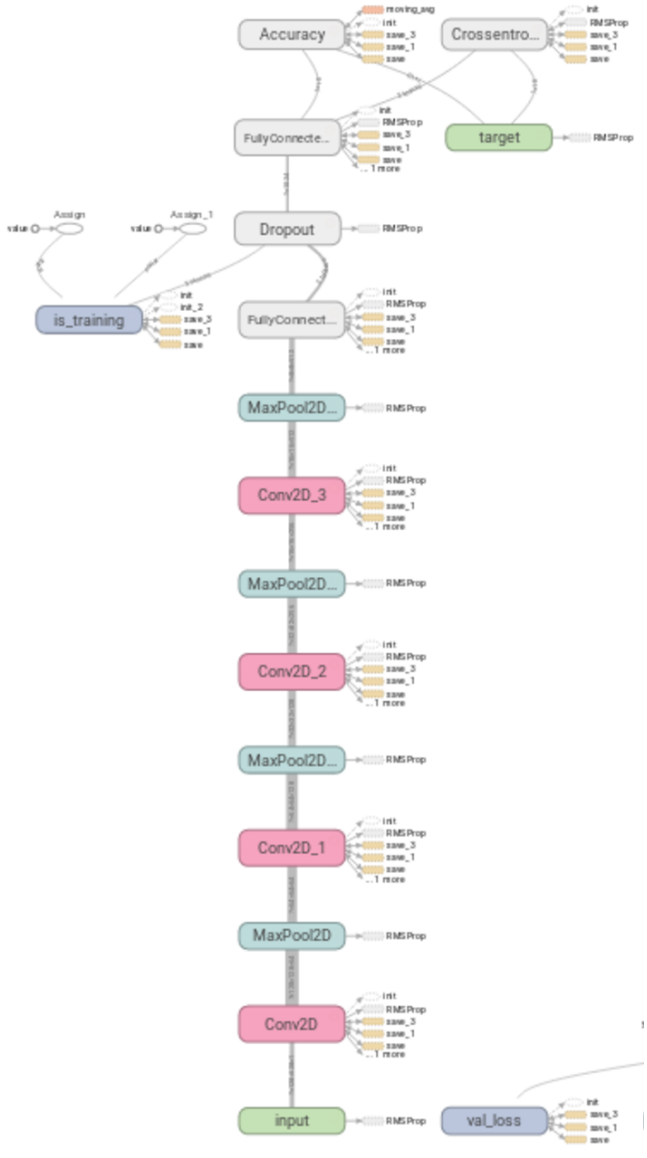

Figure 8: Our tensorflow graph

A more visual way to show this graph is used in the article (figure 9) and represent the size of the tensors flowing through the layers.

Results

The evolution of our accuracy and lost function during the 2 last training of this network (that lasts for 12 and 30h) can be seen in figure 10. We say that the training accuracy constantly increase and hasn't reached 100% yet at the end of

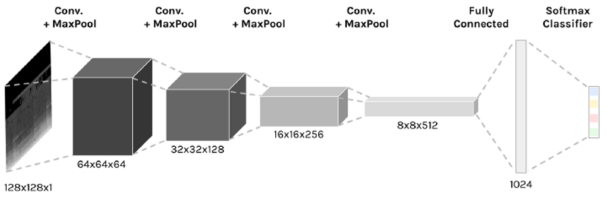

Figure 9: from Finding the genre of a song with Deep Learning — A.I. Odyssey, Julien Despois

simulation (our model is complex enough to achieve this with enough iterations). We also oberves that the validation accuracy was increasing during training but also dropping each time we change our training batch.

As we weren't able to make the program work on a supercomputer, we have not enough computation time and are not able to tell if the model was going to achieve a good result. The first results were encouraging but we can see that the loss on validation starts increasing at the end of the second simulation, it could mean that we are starting to overfit, maybe the dropout ratio of the fully connected layer is not big enough to guarantee we don't overfit training data.

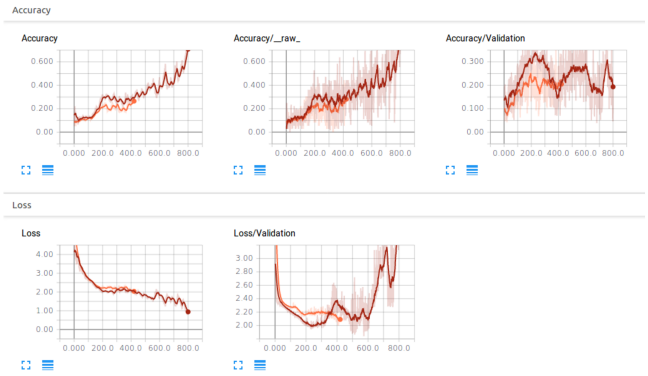

Figure 10: Evolution of accuracy and lost during training

Finally our classifier is reaching 35% to 40% of accuracy on the multiclass problem. You can see the final confusion matrix (figure 11) for the last training (30h), this matrix correspond to 37% of accuracy.

```
[[ 50  31   0   0   3   2   0   0  11   3]
 [  0 100   0   0   0   0   0   0   0   0]
 [ 45  44   0   0   0   2   0   0   2   7]
 [ 40   0   0   0   1   0  39   0   8  12]
 [ 18   0   0   0   0   0  82   0   0   0]
 [  0  49   0   0   0   0   0  43   8   0]
 [  4   0   0   0   0   0  96   0   0   0]
 [  0   0   0  13   7   0   0  53  24   3]
 [  0   0   0   4  10   0   0  11  75   0]
 [ 14   0   0   0   0   0  86   0   0   0]]
```

Figure 11: Final confusion matrix on multiclass problem

Acronyms

ACC Accuracy. 5

CNN Convolutional Neural Network. 6

DCNN Deep Convolutional Neural Network. 6

GTZAN George Tzanetakis Dataset. 3

KNN K-Nearest Neighbors. 4

LDA Linear Discriminant Analysis. 2, 4

MARSYAS Music Analysis, Retrieval and synthesis for Audio Signals. 2, 3

MFCC Mel-Frequency Cepstral Coefficients. 2

MMSE MSN Music Search Engine. 2

MPSVM Multi-category Proximal Support Vector Machine. 2

MSN Microsoft Service Network. 2, 8

PCA Principal component analysis. 3, 4

PNWC Pourcentage of Negative Well Classified. 5

PPWC Pourcentage of Positive Well Classified. 5

STFT Short Time Fourier Transform. 2, 6

SVM Support Vector Machine. 2, 8

List of Figures

List of Tables

References

[1] François Pachet - Daniel Cazaly. "A taxonomy of musical genres". In: (2000). URL: https://www.csl.sony.fr/downloads/papers/2000/pachet-riao2000.pdf.

[2] Dannenberg Roger - Foote Jonathan - Tzanetakis George - Weare Christopher. "New directions in Music Information Retrieval". In: (2001). URL: http://hdl.handle.net/2027/spo.bbp2372.2001.037.

[3] Julien Despois. *Finding the genre of a song with Deep Learning*. URL: https://chatbotslife.com/finding-the-genre-of-a-song-with-deep-learning-da8f59a61194.

[4] Li Tao - Tzanetakis George. "Factors in automatic musical genre classification of audio signals". In: (2003). URL: http://ieeexplore.ieee.org/abstract/document/1285840/authors.

[5] Professor Colin H Hansen. "FUNDAMENTALS OF ACOUSTICS". In: (). URL: http://www.who.int/occupational_health/publications/noise1.pdf.

[6] Michael Haggblade - Yang Hong - Kenny Kao. "Music Genre Classification". In: (2011). URL: http://cs229.stanford.edu/proj2011/HaggbladeHongKao-MusicGenreClassification.pdf.

[7] Merkelbach Kilian. "Feature extraction for musical genre classification". In: (2015). URL: http://hpac.rwth-aachen.de/teaching/sem-mus-15/reports/Merkelbach.pdf.