Music Genre Classification

Second Assignment of the Curricular Unit Machine Learning.
*Course Instructor

Diego Hernandez
Department of Electronics, Telecommunications. and Informatics. University of Aveiro
Aveiro, Portugal
dc.hernandez@ua.pt

Rodrigo Pereira
Department of Electronics, Telecommunications. and Informatics. University of Aveiro
Aveiro, Portugal
rodrigo.pereira@ua.pt

Prof.Petia Georgevia*
Department of Electronics, Telecommunications. and Informatics. University of Aveiro
Aveiro, Portugal
petia@ua.pt

***Abstract*—Audio data is becoming more relevant source of information in machine learning. Technologies are using more voice as a mean of interaction with smart agents, like Siri and Alexa. Music Stream Services use machine learning as a way to suggest new songs to the user by its taste of music. Music Genre Classification is one of the many tasks that can be delivered by using machine learning. In this assignment we explore different learning architectures for classifying music audio files into eight different genres. Such architectures include, convolutional and others include recurrent layers to extract features from both frequency and time dimension. We also used deep learning models. Finally we used features of the data-set's songs extracted with the librosa python library and Echo Nest/Spotify API. With such features we trained Simple Neural Networks and build Support Vector Machines to classify the genre of a song. Throughout this report we present results from the best obtained models.**

***Keywords*—CNN, CRNN, SVM, ResNet, Spectrogram, Waveform, PCA.**

I. Introduction

Music information retrieval is a field of research that has been growing over the past years. One of the many tasks is the identification of a music's genre based on the diverse features that it exhibits.

With the sudden rise of Machine learning over the past years enabled by deep neural network models, researchers have been trying to obtain better results on the mentioned task, than the previous methods could.

It was in the light of exploring this research field that we conducted our work. Our primary objective being the discovery and analysis of the state of the art architectures and solutions that have been created for this task. As well as explore if this are in fact viable solutions for enterprise and commercial use, or if they still inflexible and very research oriented.

II. State-of-the-Art

As mentioned in the introduction our work was heavily based on the recent research and novel architectures created to address this task.

A. End-to-end learning for music audio

In the paper "End-to-end learning for music audio" [4] published in 2014, the authors note that typical approaches to the problem that are based on feature engineering and shallow and simple network architectures are outdated for the problem in question. Given this fact they propose and explore three main approaches to the problem, from which we mainly explore and test two.

The first is the use of 2-dimensional CNNs to extract features from the Mel-Spectrogram of the excerpt of audio in question. The Mel-Spectrogram is obtained through the "overlapping" of the Fast Fourier Transforms (mechanism that allows to analyze the frequency content of a signal), and that end up representing the signal's loudness, or amplitude, as it varies over time at different frequencies (see Figure 1).

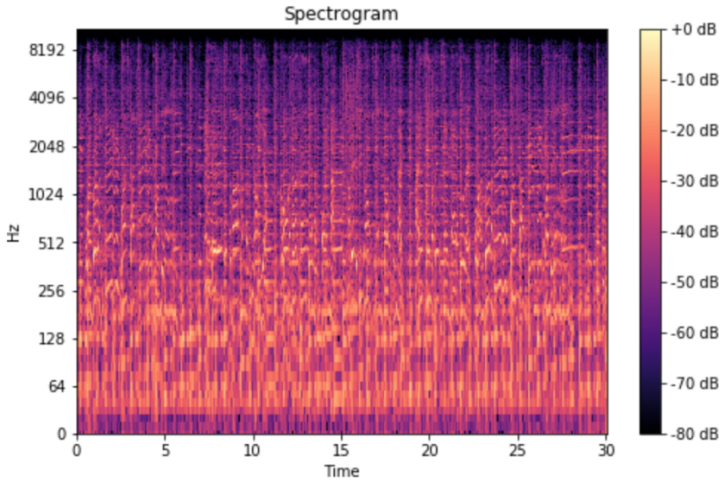

Fig. 1. Visual representation of a mel-spectrogram

The second approach would be to take in as input the raw waveform (Figure 2) of an audio sample and feed it to a stack of one-dimensional convolutional networks. However this approach comes with a difficult problem to overcome, and that is the fact that raw Waveforms come in the shape

of very large vectors, since that to ensure definition they are usually sampled at around 22kHz, which translates to a vector of 220050 values in an music excerpt of 10 seconds. That means that passing this enormous vector to a stack of "full convolutional layers"(stride equals to 1) would be computationally too expensive. As such the authors propose to make the first convolutional layer to be strided, in order to reduce the magnitude of the input going into the following fill convolutional layers ahead.

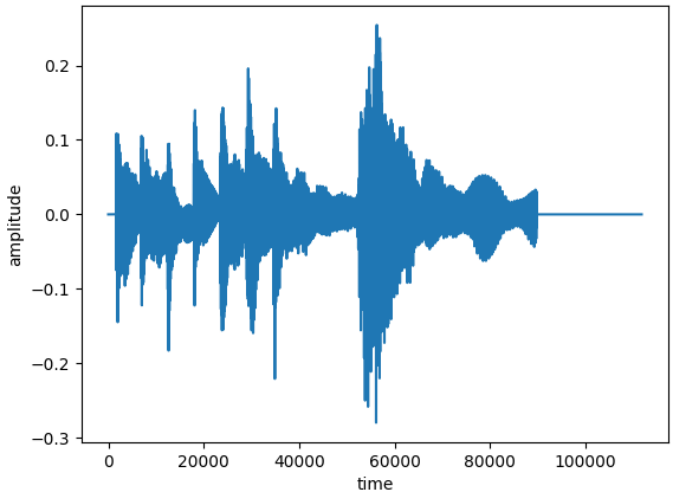

Fig. 2. Waveform

Comparing both these networks the authors obtained decent values of performance for the spectrogram case, but not so good values for the raw waveform case. They hypothesize their network to still be to simple, given the huge input they are feeding it.

In the tentative to improve on this model, authors of the paper [6] propose a network similar to theirs, with the difference that, inspired by popular architectures such as VGG, they use very small features (which also makes the network deeper). The network built ended up obtaining great results for the Million Song Dataset (MSD) [1], competing with other state of the art spectrogram architectures.

We also explored the work done by Keunwoo Choi, Gyorgy Fazekas and Mark Sandler on the paper "Convolutional Recurrent Neural Networks for Music Classification" [2] in which the authors propose the use of an Long Short Term Memory [LSTM] that is fed by the outputs of some prior 1-dimensional Convolutional networks in order for the network to have some memory of the music structure overall (Figure 3).

It was mainly on these set of works that we based our approaches to the problem, but is also worth mentioning that there are other works on the field with great performance results such as ReSE-2-Multi [5] , Timbre CNNs [8] and others.

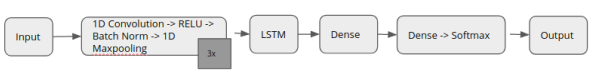

Fig. 3. CRNN Model (credit: https://towardsdatascience.com/using-cnns-and-rnns-for-music-genre-recognition-2435fb2ed6af)

III. Resources

To Properly train a deep neural network in a very restrict and limited time window it is essential to have great computational resources such as processors and GPU to handle skillfully the the consequent complex and heavy computational tasks.

We decided to go with the usage of Google Colab (https://colab.research.google.com/notebooks/intro.ipynb) application to train our models. It is a cloud service allowing AI developers to train neural network models on a Tesla K80 GPU for free, allowing to handle efficiently those highly complex computational tasks. It also allows each user to have the usage of almost 12 GB of RAM and 110 GB of memory in Disk.

Taking advantage of this application leaded to fasten the training of our models,which wouldn't be possible efficiently with GPU and processors at our disposal. However we noticed a limitation, which is: Colab is able to provide free resources in part by having dynamic usage limits that sometimes fluctuate, and by not providing guaranteed or unlimited resources. So in some occasions we weren't able to use GPU because of a overused GPU limitation.

For more information about Google Colab and limitations: https://research.google.com/colaboratory/faq.html. .

IV. DataSet

A. Data Retrieval

As our data-set, we used FMA: A Data-set For Music Analysis [3]. It is an open and easily accessible data-set suitable for evaluating several tasks in MIR. It provides full-length and high-quality audio, pre-computed features, together with track and user-level metadata, tags, and free-form text such as biographies. Furthermore it is balanced (Figure 4)

There were plenty of music data-set available online. However, we didn't have great computational resources to store and process such data-sets with significant sizes. We ended up choosing FMA dataset for the main reason that it has available various sizes of MP3-encoded audio data. The chosen one is the smallest one, a data-set of 7.2GB with 8000 tracks of 30s, with 8 balanced genres.

B. Data Pre-processing

Besides the Music Metadata, we wanted to use data from the music itself. Throughout this assignment it has been used models which inputs were information belonging from waveforms, spectrograms and images of the spectrograms.

To load and produce waveforms and spectrograms of each song, we used librosa [7]. It's a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.

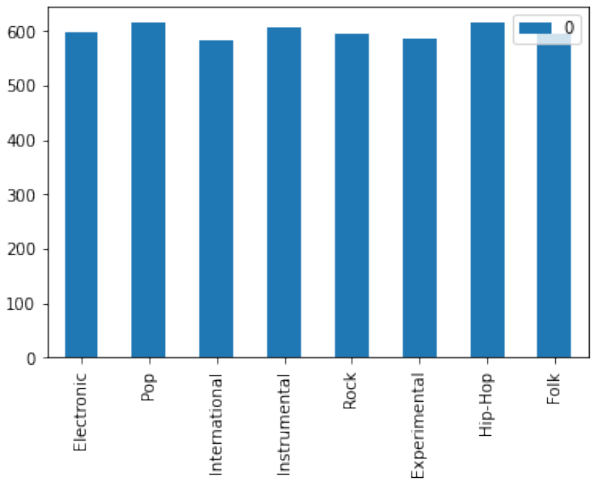

Fig. 4. Number of examples by label

It takes nearly a second to load a song and retrieve the information of its corresponding waveforms and spectrograms. Furthermore it isn't feasible to load 8000 music in 2 hours to later train our models.

To avoid the mentioned processing time, we write information from the songs' waveforms, spectrograms to pickle files.

By self-experience, we thought that humans don't usually take more than 5-10 seconds to determine the genre of a song. In [6] having just music samples of approximately 2.5 seconds of duration, they have achieved great models of genre classification with accuracy ranging from 80% and 90%. Moreover,it becomes more flexible to split the song into smaller windows, because when loaded for the training and validation of our models, it will consume less RAM. Hence, we decided to split the images, with a sample window of 10 seconds.

During the writing process, it was noticed that some songs doesn't have the same amount of information as the one expected. For example, generally, when loaded, the shape of a spectrogram of our samples is (128, 388). However not all of them has the expected shape or quantity of information. We assured the filtration of these cases, along with some musics with silence for over a second.

Although the reduction of the duration of the songs, the amount of music information that is loaded, for training our models, surpassed the limitations of the available RAM capacity. That being said, we did proceed and developed our own data generator callbacks. Hence during the training of out model, every time it needs a new batch to train and validate the model, it uses the callback to retrieve a new batch, only loading a small fraction of the data-set sequentially during the training and validation-phase.

Music Metadata FMA Dataset provides metadata of each available song.It contains features of each song extracted with librosa and audio features provided by Spotify API.

Genre	N° **songs**
Pop	1000
Folk	1000
Rock	1000
International	1000
Instrumental	1000
Hip-Hop	1000
Experimental	1000
Electronic	1000
Set Type	N° **songs**
training	6400
test	800
validation	800

TABLE I
LIBROSA DATASET DESCRIPTION

1) Librosa: With librosa it was possible the extraction of a total of 518 features for each song. Some relevant features are:

- **Chroma CENS**: Chroma variant Chroma Energy Normalized.
- **Chroma CQT**: Constant-Q chromagram.
- **Chroma STFT**: chromagram from a waveform or power spectrogram.
- **MFCC**:the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.
- **Spectral Rolloff**:The roll-off frequency is defined for each frame as the center frequency for a spectrogram bin such that at least 0.85 of the energy of the spectrum in this frame is contained in this bin and the bins below.
- **Tonnetz**: A conceptual lattice diagram representing tonal space

We used a balanced data set of a total of 8000 song.The data-set is split into training, validation and testing sets. Detailed information about the data-set obtained with librosa is shown in table I

Each set has songs belonging to 8 genres, distributed in a balanced manner. This is, the number of songs for each genre is equal in its corresponding set.

2) Spotify API: With the Echo Nest/Spotify API it was possible to extract a total of 249 features for each song. Each song contains features such as: Acoustiness, Danceability, Energy, Instrumentalness, Liveness, Speechiness, Tempo, Valence, and Temporal Features. This features are relevant to music stream services such as Spotify, in order to cluster similar songs and those can be shared with users based on their preference, for example.

With such features it could become more feasible to train better a model capable of classifying the genre of the music.

We used a balanced data set of a total from 1920 song. The Data-set is split into training, validation and testing sets. Detailed information about the data-set obtained with Echoest/Spotify API is shown in the table II

Each set has songs belonging to 8 genres, distributed in a balanced manner. This is, the number of songs for each genre

Genre	N° songs
Pop	240
Folk	240
Rock	240
International	240
Instrumental	240
Hip-Hop	240
Experimental	240
Electronic	240
Set Type	N° **songs**
training	1152
test	384
validation	384

TABLE II
SPOTIFY API/ECHONEST DATASET DESCRIPTION

is equal in its corresponding set.

To notice that it wasn't possible to extract features of all 8000 songs using Echo Nest, since Spotify service does not have available all these songs belonging to the data-set. Taking into account all the available songs by the service, and in order to make our data-set balanced we had to select a maximum number of 240 songs for each of the 8 genres.

C. Waveform

In electronics, acoustics, and related fields, the waveform of a signal is the shape of its graph as a function of time, independent of its time and magnitude scales and of any displacement in time. The waveform of a steady periodic sound affects its timbre. Each song can have complicated and distinct Waveforms. An example of a waveform is illustrated in figure 2.

D. Mel Spectrogram

To construct a Mel Spectogram the following need to be done: Digitally represent an audio signal. Map that signal from the time domain frequency domain using fast Fourier transform, performing this on overlapping windowed segments of the audio signal. It is converted the y-axis, representing frequency, to a log scale and the color dimension, representing amplitude, to decibels to form a spectrogram. Hence, a spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. It is mapped the y-axis, corresponding to frequency, onto the mel scale to form the spectrogram. A mel scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another.

All these above mentioned operations are easily established using librosa library.

An example of a mel spectrogram is illustrated in figure 1

V. MACHINE LEARNING METHODS USED

A. Simple 2D CNN on Spectrogram Data

One approach we decided to make was to use a very basic and small network consisting of two to four convolutional layers.

We build an iterative constructor of the network that allowed us to vary different parameters of this network, in order to find the model which gave the best performance.

All trained convolutional models performed bad, with low accuracy and significant loss for the validation set. One of the models architecture is shown in figure 7 and its accuracy and loss results through each epoch using training and validation set shown in figure 8

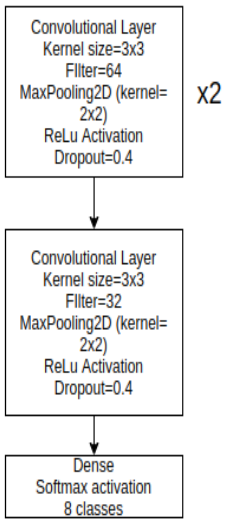

Fig. 5. CNN Architecture Model trained with Spectrograms data.

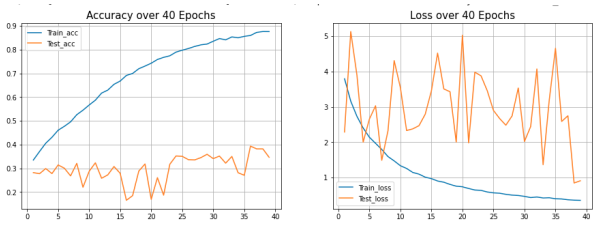

Fig. 6. Training and validation accuracy of CNN trained with Spectrograms data.

As we can see in the figures mentioned before, the model was over fitting, like the other CNN models trained for this assignment. A way to tackle this problem may be to have a more complex network and a much bigger data-set, so the model can focus/understand different features from the spectrogram, and thus have better results and a more robust classifier.

Considering all the mentioned during this section, we moved on to different and more complex architectures.

B. CRNN

As mentioned in state of the art section, this architecture makes use of an LSTM after the convolutional layers in order to get an idea of the music structure (given that for that purpose it is necessary to have "memory" hence the use of an LSTM).

As such we built our model following the same philosophy of having an iterative model builder that tested different variations of the network such as the number of convolutional

layers, the number of filters, the filter size, and the pooling sizes of the MaxPooling layer.

The architecture of our best model is shown in figure 7. For 50 epochs the training and validation accuracy and loss are illustrated in the graphs of figure 8.

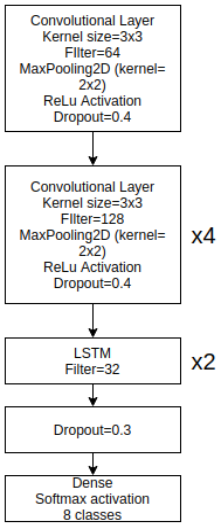

Fig. 7. Best CRNN Architecture Model

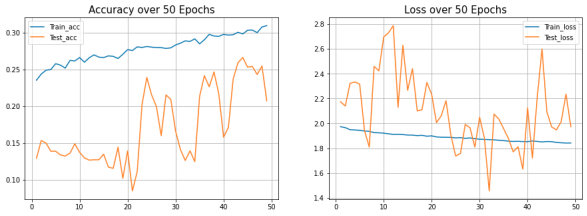

Fig. 8. Training and validation accuracy of CRNN using data from the spectrograms.

Looking at figure 7 we see that this model needs more training, in other words more epochs. However we can see that it the model will tend to suffer from a over fitting behaviour, since the validation accuracy and validation loss diverge significantly, for better (validation accuracy increases,validation loss decreases) and also for worse(validation accuracy decreases,validation loss increases), between epochs. So even with this setup we weren't getting anywhere, at least with the given epochs, our loss was high and our accuracy very low. So we kept trying more complex models to fit or data.

C. SimpleCNN

To combat the underfitting problem we were having, we decided to go deeper, and so after studying the studying the SampleCNN paper [6], we decided to replicate their architecture and apply it to our data. This model worked with raw waveforms as mentioned before. We went with the 3^9 architecture (Figure9), this means the network we used had 9 non strided convolutional layers (excluding the last/prediction one) and the stride of the first one is 3. Having this configuration leads us to make the input to be equal to $(3 * 3^9)$ (stride size of first convolution times stride size of first convolution number of layers to the power of the number of layers), which equals 59049 samples (2678 ms of music with a 22kHz sampling rate). This network has this very specific structure and input size in order to address log-scale amplitude compression and phase-invariance, that where problems pointed to the model on the paper [4](the explanation for this fact is explained in more detail on the original paper [6]).

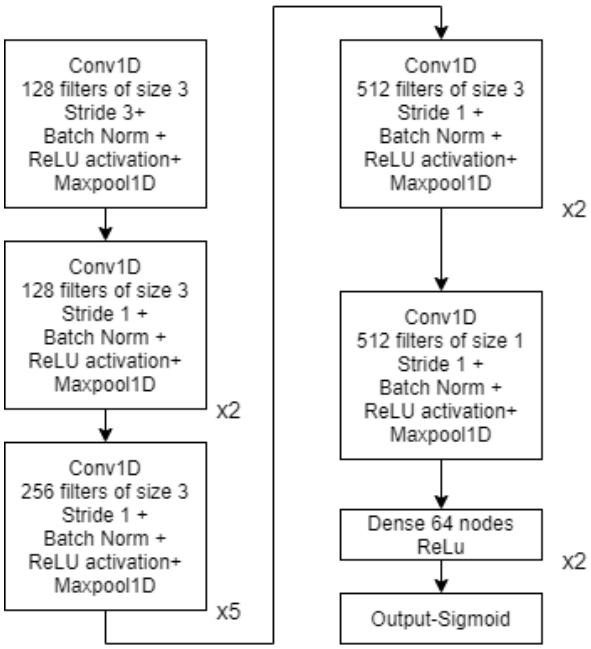

Fig. 9. SampleCNN 3^9 architecture

So in order to apply this model we readjusted our data-set, built the model, and started training, this time we didn't use the approach of iteratively try new configurations, since we wanted to be as true as possible to the original architecture, so we only tuned aspects such as learning rates dropout rates, batch sizes and epochs trained.

To our surprise the model preformed very badly leaving us with a 12% validation accuracy and very high loss as shown in the following figures:

The model was clearly overfitting, to combat that issue, we tried adding dropout layers (Which proved to not have a significant effect, just delaying the overfitting exponential a few epochs). We came to the conclusion that we did not have enough data for this model (since the original work was tested on the million sound dataset [1]). We tried to duplicate our dataset by using different parts of the music as different songs, but it proved to not matter, it was still too little data.

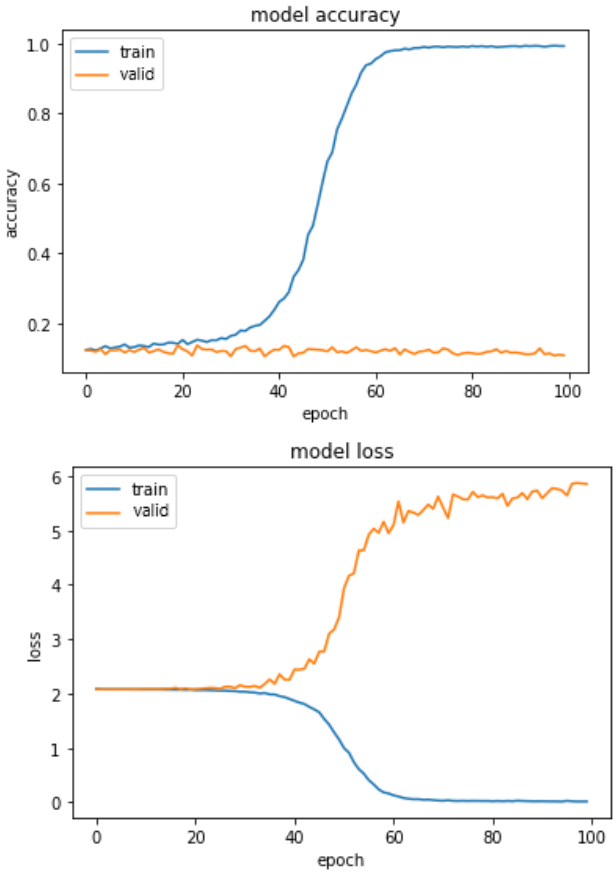

Fig. 10. Loss and accuracy over the epochs on the SampleCNN model

D. ResNet50 with images

Since we weren't getting results with the previous approaches, and since on our last project we fount success with ResNet50, we decided to save all spectrograms as PNGs and feed them to a ResNet50 pre-trained with ImageNet.

This despite still being slow to train the model (most likely due to our image generator, and not due to the network per say), was showing better results than the previous models, reaching around 50-60% validation accuracy, which puts us on the level of some kaggle projects, but nowhere as near as the papers that where using a much larger dataset.

Figure 11 shows different models we trained searching for the best one.

E. Suport Verctor Machine

We used Support Vector Machine and used it with the librosa dataset. The results are shown in table III

We also applied Dimension reduction of the features taken from librosa applying Principal Component Analysis (PCA) with a variance equal to 90%. The original features passed from 518 features to 151 features.III. After that we build the SVM model and the following results were the shown in table IV

Although the contained accuracy was slightly lower than the accomplished in SVM without dimension reduction, it was possible to reduce a significant amount of data and obtain similar accuracy. This behaviour is noticeable in the following algorithms/models described below.

However, both results are not desirable, both having accuracy under 50%. We proceeded by building Simple Neural

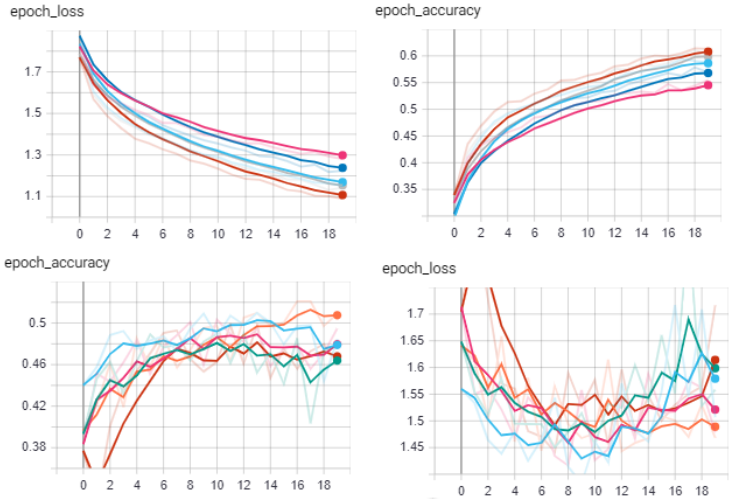

Fig. 11. Train (Upper images) and Validation (Lower images) accuracy and loss over the epochs

SVM using features from librosa				
Genre	Precision	Recall	F1-score	Support
Electronic	0.60	0.64	0.62	100
Experimental	0.42	0.43	0.42	100
Folk	0.24	0.23	0.23	100
Hip-Hop	0.68	0.73	0.70	100
Instrumental	0.45	0.45	0.45	100
International	0.57	0.46	0.51	100
Pop	0.39	0.39	0.39	100
Rock	0.61	0.63	0.62	100
accuracy			0.49	800
macro avg	0.49	0.50	0.49	800
weighted avg	0.49	0.49	0.49	800

TABLE III
SVM RESULTS

SVM using features from librosa				
Genre	Precision	Recall	F1-score	Support
Electronic	0.56	0.62	0.59	100
Experimental	0.43	0.42	0.43	100
Folk	0.22	0.22	0.22	100
Hip-Hop	0.66	0.71	0.69	100
Instrumental	0.43	0.46	0.44	100
International	0.58	0.46	0.51	100
Pop	0.35	0.35	0.35	100
Rock	0.64	0.63	0.63	100
accuracy			0.48	800
macro avg	0.48	0.48	0.48	800
weighted avg	0.48	0.48	0.48	800

TABLE IV
SVM RESULTS

Metric	Value
Loss	1.5184
Accuracy	0.4912
Precision	0.0292
Recall	0.5984
F1 Score	0.3930

TABLE V
METRICS RESULTS OBTAINED IN SIMPLE NEURAL NETWORK USING FEATURES TAKEN FROM LIBROSA

Networks composed of Dense Layers and Dropout Layers.

We built our model following the same philosophy of having an iterative model builder that tested different variations of the network suck as the number of dense layers, the number of filters, different regularization values and drop out rates.

The best result we have obtained was with a Neural Network with two dense layers of 32 and 16 outputs, sequentially, a dropout rate out 50% after each Dense layer and a learning rate value of 0.00025.

In figure 12 we can see the development of the model by training it with the data features retrieved from librosa.

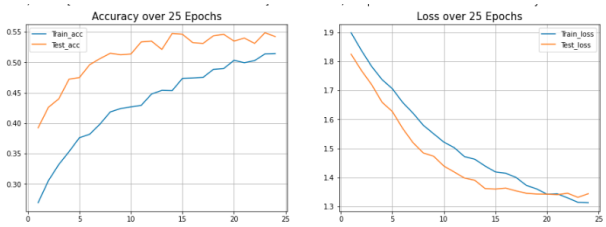

Fig. 12. Training and validation accuracy of simple Neural Network model along the epochs using librosa data.

As seen in table V, the values are still not acceptable, having low values of Accuracy, Precision, Recall and F1-Score. Moreover, the loss is significant.

As an next step, we did the same approach already mentioned in this section, and the only difference is that we used instead data features obtained by Echonest/Spotify API. The obtained results are pointed out in the notebook SVMNN.ipynb

Overall, the best obtained result was from a SVM developed along with the Spotify API data-set, without using dimensional reduction (This is, without using PCA). The results are shown in table VI

So we can see that Spotify API Dataset comes as a data with more useful information than the data retrieved from librosa, letting us have achieved a SVM model capable of classifying a song's genre from the 8 genres shown in the table VI, with an accuracy of 62%

Finally, for the songs whom features are registered in the considered Spotify API Dataset, we appended the features that belong to the librosa dataset, appending the feature values to each corresponding song. In the end we would construct SVM models and Neural Network Models from a Dataset with 1152 records of 750 features.

SVM using features from librosa				
Genre	Precision	Recall	F1-score	Support
Classical	0.87	0.94	0.90	48
Electronic	0.67	0.46	0.54	48
Folk	0.41	0.69	0.51	48
Hip-Hop	0.67	0.77	0.72	48
Jazz	0.63	0.46	0.53	48
Old-Time / Historic	0.98	0.96	0.97	48
Pop	0.29	0.33	0.31	48
Rock	0.68	0.35	0.47	48
accuracy			0.62	384
macro avg	0.65	0.62	0.62	384
weighted avg	0.65	0.62	0.62	384

TABLE VI
SVM USING SPOTIFY API DATASET

SVM using features from librosa				
Genre	Precision	Recall	F1-score	Support
Classical	0.98	0.96	0.97	48
Electronic	0.69	0.42	0.52	48
Folk	0.49	0.73	0.59	48
Hip-Hop	0.66	0.81	0.73	48
Jazz	0.72	0.58	0.64	48
Old-Time / Historic	0.98	1.00	0.99	48
Pop	0.30	0.40	0.34	48
Rock	0.81	0.44	0.57	48
accuracy			0.67	384
macro avg	0.70	0.67	0.67	384
weighted avg	0.70	0.67	0.67	384

TABLE VII
SVM RESULTS USING FEATURES DERIVATIVE FROM LIBROSA AND SPOTIFY API

However, between the trained neural network models and SVM, the best results we have got was from a SVM build with a resulting dataset derivative from the appliance of PCA, with 90% of variance, to the above mentioned data-set, resulting of a dataframe of records with 201 features.

The results are shown in table VII, and confusion matrix obtained is shown in figure

Finally, between all the build models, the one who has proven to be the best classifier of songs' music genre was the SVM model build along with the features taken from librosa and Echo Nest/Spotify API.

VI. OVERALL RESULTS AND DISCUSSION

Overall we felt like our biggest obstacle in realizing this project was the inability of having a large dataset and great computational resources. This is due to the fact that the data we were dealing were very complex to simple models to grasp, but as the same time having struggling with bigger and complex models, since they required a large data-set in order to not overfit.

We also expected the ResNet to preform a bit better, but we are not so surprised it gave the results it did, since the data which had been pre-trained on is very different from the one we fed them. Despite all that, we ended up pleasantly surprised at the performance that using simple models (Neural

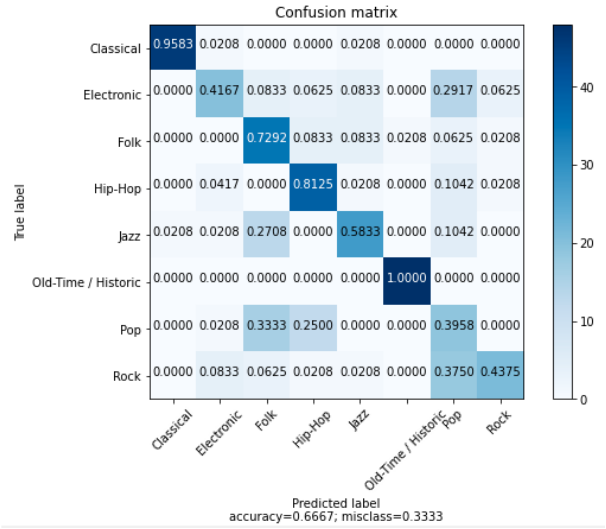

Fig. 13. Confusion Matrix corresponding to the results obtained at the test face of the SVM build with the dataset derived from librosa and echonest data-set

Networks and SVM) had in relation to all of this complex models, proving that complex and top of the line models are not always the best choice for all situations.

VII. Conclusion

After implementing all of this models and as a conclusion to our initial premise, we came to the conclusion that, despite the recent surge in deep neural networks applied to music information retrieval field, it is still much of a niche and newborn concept compared to Natural Language Processing (NLP) or Computer Vision. It is evident this technologies are not ready to go onto the commercial level of machine learning, since the accuracy results obtained require too large datasets, very heavy computational power, and also due to the fact that there yet to be produced high quality, and high standardized pre-trained models, that could be used by anyone wanting do do some project based on the concept, without having a high end machine learning team, as well as the capacity of having high computational resources. In future work we would ideally scale up our dataset (probably to something like the Million Song Dataset), retest our previous models for that dataset. Research more architectures that could help us improve in terms of performance. And lastly explore the possibility of taking the concatenation of the output of 2 models(that use different types of input eg. waveform + spectrograms) as an input to another network, to see if this would help the performance.

References

[1] Thierry Bertin-Mahieux, Daniel Ellis, Brian Whitman, and Paul Lamere. The million song dataset. pages 591–596, 01 2011.

[2] Keunwoo Choi, György Fazekas, Mark B. Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. *CoRR*, abs/1609.04243, 2016.

[3] Michaël Defferrard, Kirell Benzi, Pierre Vand ergheynst, and Xavier Bresson. FMA: A Dataset For Music Analysis. *arXiv e-prints*, page arXiv:1612.01840, December 2016.

[4] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968, 2014.

[5] Jongpil Lee, Taejun Kim, Jiyoung Park, and Juhan Nam. Raw waveform-based audio classification using sample-level CNN architectures. *CoRR*, abs/1712.00866, 2017.

[6] Jongpil Lee, Jiyoung Park, Keunhyoung Kim, and Juhan Nam. Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification. *Applied Sciences*, 8(1):150, Jan 2018.

[7] Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. pages 18–24, 01 2015.

[8] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. *CoRR*, abs/1703.06697, 2017.