Music Genre Classification using State-of-the-Art Deep Learning Techniques and Novel Data Processing Methods

Marc Watine, Auguste Lefevre, Lukas Mautner
Deep Learning Project
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—The goal of this Deep Learning end-of-semester project was to (a) push the general performance of music genre recognition forward and (b) introduce a new method for pre-processing which allows for faster experimentation and model tuning in the future. We experimented with two different musical representations: mel-spectrograms, extracted from the musical signal itself into images capturing the frequency spectrum at each given point in time and manually extracted features. Using the mel-spectrograms we applied a *Divide and Conquer* approach, fragmenting them into multiple smaller chunks, which gave us the best results and reduced the number of parameters by a factor 5 to 30. For the manually extracted features we lowered the dimensionality of the data by a factor 10 - using a relevant latent space representation based on an deep auto-encoder approach - and achieved the highest accuracy compared to other representations in higher dimensions.**

I. Introduction

A. Motivation

Music Information Retrieval (MIR) is a major field of study within machine intelligence, with many as of yet unsolved challenges [1]. Among music enthusiasts and casual listeners alike, the genre of a song plays an important role. Many decisions about whether to include a particular piece of music in a playlist can be made solely based on the genre – if it is indeed accurate. As a consequence, correctly predicting the genre of a song allows one to set up a number of convenience systems around music: recommenders, sorters, automatic playlist continuation and so on. This is especially relevant to music streaming platforms, as the number of tracks offered there continues to grow[1]. Motivated by this fact, we set out to address one of the challenges in MIR: Music Genre Recognition (MGR).

B. Related Work

Historically, most MGR solutions used various methods to extract and select features that they would then feed to classifiers [2] [3]. Among these features, Mel-Frequency Cepstral Coefficients (MFCCs) are usually predominant [4]. MFCCs are short-term spectral-based features relying on a *mel-scale*, which is based on a smoothed mapping between actual frequencies and human-perceived pitch [4]. Among the classifiers used with these MFCC features are deep learning (DL) models [5]. The latter have also been used in an attempt to solve the feature selection problem [6].

As the popularity of convolutional neural networks (CNNs) grew, so did their use for MGR (see [7] and [8]). CNNs are inherently useful for the MGR problem since they learn hierarchical features over multi-layer structures and genre tags can be seen as high-level features. A time-frequency representation of audio is often used in the form of a so-called *Mel Spectrogram* [9]. More recently, CNNs have been combined with recurrent layers to form R-CNNs, which has significantly improved the performance [10]. New methods build on top of CRNN such as a Divide and Conquer approach as been recently mentioned in an article [11]. On the other hand, manually extracted features have already proven their efficiency and limits. However, research on future selection such as [12] led us to pursue the work in this direction by trying to represent the feature in a latent space using state of the art method such as auto-encoder in order to optimize the prediction as well as the computational resources needed for features based MGR.

II. Data

The main challenges in assembling an appropriate dataset for this task are (a) copyright laws, due to which we can only use songs under the Creative Commons license, and (b) inherent label noise, which stems from the fact that many songs belong to more than one genre.

A. Source Dataset

The GTZAN dataset [13] is the most well known dataset in MIR and has often been used as a baseline ([2], [3], [14]). Despite its popularity, we decided not to use it since (a) it contains only 10 genres and is balanced, which does not reflect the real world, and (b) it only counts 100 samples per genre. We decided to use a more recent, large-scale reference dataset: the Free Music Archive (FMA) [15].

It counts $106'574$ tracks, with $16'341$ artists arranged in a hierarchical taxonomy of 161 genres. In addition to the high-quality audio, it provides pre-computed features and track metadata. FMA also provides a pre-defined split into training/validation/test, since doing that from scratch is

[1]Spotify (the most popular streaming site, having more than 450 million users) currently counts more than 80 million tracks available.

highly non-trivial due to the multitude of possible dependencies (artists, albums, collaborations, etc.). For computational reasons, we decided to use the *medium* version of the dataset, which only counts $25'000$ tracks of 30s. The class imbalance is visualized in figure 3.

B. Mel-Spectrograms

A musical signal can be transformed in many ways to be fed to trainable models. As mentioned in section I-B, we will work with the Mel-Spectrograms of the individual tracks. You can find a visualization of mel-spectrograms in figure 4. Using the *Librosa* library [16], a mel-scaled power spectrogram with 128 bands is computed for each segment - 30s, 10s and 3s-. We then applied a log-scale and a min-max normalisation (See II-B1).

1) Data Pre-processing: Our starting point is a set of tracks of 30sec (raw audio). As mentioned earlier, we want to transform the raw audio into mel-spectrograms. We also want to divide each 30sec raw audio into $3 \times 10sec$ and $10 \times 3sec$ samples in order to perform the Divide and Conquer approach later. The pre-processing pipeline can be described as follow:

- capture the raw audio using *librosa.load()*
- ensure that the length of the audio is 30sec by applying a padding method
- cut the 30sec audio in $3 \times 10sec$ and $10 \times 3sec$ snippets
- compute mel-spectrograms of each snippets using *librosa.feature.melspectrogram()* with standard 22'050 Hz sampling rate and setting the FFT's parameters to 2048 window size and a hop size of 512
- convert power spectrograms (amplitude squared) to decibel (dB) units using *librosa.power_to_db()*

The last step is the normalization of the data. Based on related work I-B, we apply a min max normalization:

$$\tilde{x}_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}. \quad (1)$$

C. Manual Features

The Librosa library was used to extract a total of 518 manual features for each track (among them are also the MFCCs). Since these manual features each have meanings of their own but do not necessarily relate to each other, we will explore their predictive powers and redundancies in section III-D.

III. Models & Methods

A. Strategy Overview

First, we will work on the state-of-the-art models classifying mel-spectrograms (see [10]), setting-up a drastic reduction of the trainable parameters. In particular, we will take a *divide and conquer* approach mentioned in I-B.

Secondly, we also use the manual features provided by FMA to perform MGR. Here, we focus mainly on reducing the dimensionality of these features using both established methods and our own deep auto-encoder. We do this motivated by [12], which demonstrated the feasibility and usefulness of such an approach.

B. Mel-Spectrogram

For the Mel-Spectrogram strategy, we first implemented a LeNet-5 [17] architecture as a baseline. Unsurprisingly, due to the fact that LeNet was designed for lower resolution images [17], it did not yield results worth mentioning. Larger CNNs lead to an improved but not satisfactory performance. In the end, we were able to achieve desirable results with an R-CNN, the architecture of which can be seen in figure 1.

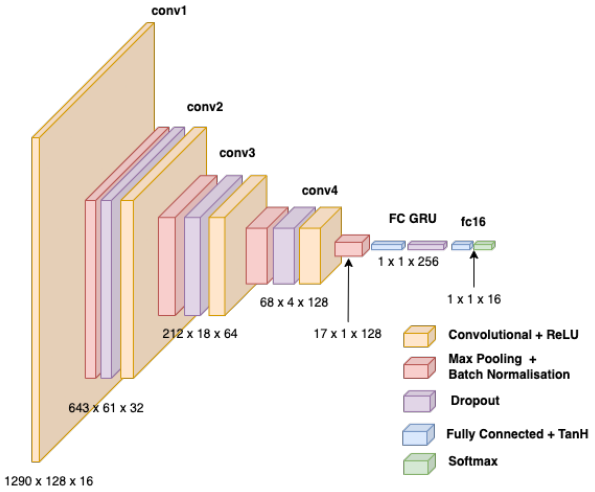

Figure 1: Base R-CNN architecture defined for the 30 second track samples

Recurrent Convolutional Neural Networks (R-CNNs) [10] combine the image convolution-pooling advantages of CNNs with the fully connected time sensitive advantages of Recurrent Neural Networks. They have proven to work especially well with tasks involving spectrograms, as they are able to capture the local patterns (convolutions) as well as temporal patterns (recurrent) (see [10] and [18]). For reasons of computational efficiency [19] and generalization [20], we opted to utilize gated recurrent units (GRUs) instead of the often-used LSTM cells within our R-CNN.

C. Ensemble Learning

One of the goals of this project was to assess the effectiveness of a divide-and-conquer approach in solving the MGR task. Specifically, we wanted to investigate the usefulness of training a larger number of smaller models with fewer parameters as compared to one large model.

For the Mel-Spectrogram strategy, we trained 3 separate types of models, for each 30s, 10s and 3s track lenghts. We then compared the performance of the single 30s model predictions to that of each ensemble voting of the 10s an 3s models. We discuss the results of this in section IV.

D. Manual Features

While selecting a useful subset of the original 518 features is beneficial in theory, doing so in a naive way is not possible since there are $2^5 18 \approx 8.6 \cdot 10^1 55$ possible subsets. In order to select intelligently, we first tried two baseline approaches (both non-deep) to verify the validity of our assumptions around the usefulness dimensionality reduction, and then used a deep auto-encoder as our actual solution.

1) Baseline 1 – SVD: We used SVD[2] to reduce the number of features per music track and then trained a Support Vector Classifier (SVC) to do the actual classification. See section IV for the list of dimensionalities we used.

2) Baseline 2 – Random-Forest Feature Importance: We used the feature-importance ranking built into the *Random Forest* ensemble method to select the n most important features after training it with the full 518 features. Like in section III-D1, we trained an SVC for the actual classification, using the reduced number of features.

3) Our Method – Deep Auto-Encoder: We used a deep auto-encoder, with the latent space being used for the classification. An illustration of the architecture is shown in figure 5. Since the Librosa features do not warrant the use of any special type of network (like a CNN for images), we used simple feed-forward neural networks for the encoder, decoder and classifier. Through the dual loss (reconstruction and classification), we hope to obtain a latent space that compresses the original features in a way that retains only those features that are important for separating one genre from another. Specifically, our aim is to produce a system that allows for much more effective compression of the feature vector (i.e. more compression while maintaining the performance) than a naive compression/selection approach.

IV. Experiments & Results

All experiments presented here can be reproduced by following the instructions in our GitHub repository[3].

A. Experiments: Mel-Spectrogram

- The so called **Base Model** is based on the original CRNN paper[10], combined with the model used on the article [11]. Refer to Fig. 1 for a complete architecture
- The **L2Reg Model** removes one of the fully connected GRU units and incorporates weight decay regularization. From this point onwards, the model is optimized on the convolutional filters, kernels, and pooling sizes.
- The **LRS Model** incorporates a Learning Rate (LRS), with the dropout but without weight decay.
- The **3Conv Model** removes 1 convolutional layer in pursuit of reducing the number of trainable parameters.
- **2Conv Model** removes an additional conv. layer.
- The **Optim Model** is the best model for each track length, combining LRS, L2 and the optimal number of convolution block

[2]https://en.wikipedia.org/wiki/Singular_value_decomposition

[3]https://github.com/AugusteLef/MGC-2022

B. Results: Mel-Spectrogram

The accuracies given are an average of multiple experiment runs, for robustness. For the 10- and 3-second models, they are computed using the ensemble majority voting **(v)** described in Section III-C. One may observe that the ensemble majority voting always perform better than the version without it and also that it allows to outperform 30sec based models most of the time.

	% Accuracy					# of
	30s	**10s(v)**	**3s(v)**	**10s**	**3s**	**param**
Base Model	59.52	62.44	64.65	61.67	57.35	807k
L2Reg Model	62.52	62.36	64.50	60.48	60.73	412k
LRS Model	62.09	63.80	65.87	61.78	61.00	412k
3Conv Model	59.14	64.81	65.44	62.23	61.07	108k
2Conv Model	57.46	60.57	62.13	58.76	58.92	26k
Optim Models	**63.06**	**64.93**	**66.14**	62.44	64.65	–

Table I: Accuracy in % and number of parameters for each model on the 6 different model architectures described in section IV-A.

C. Results: Manual Features

First, we evaluated our baseline methods: The results can be seen in table II.

	Accuracy		Time (sec)	
Dim	**Random Forest**	**SVD**	**Random Forest**	**SVD**
10	51.42%	50.78%	208.9	90.6
50	59.36%	60.27%	206.7	89.1
100	62.21%	62.43%	228.3	116.6
200	64.47%	63.87%	309.0	200.3
400	**64.57%**	**64.05%**	578.6	471.5
518	63.95%	63.93%	697.8	591.0

Table II: Accuracies and times taken for training and prediction for the baselines described in sections III-D1 and III-D2, for all tested dimensionalities.

The accuracies for our deep auto-encoder can be seen in table III.

Dim	**Accuracy**
10	62.6%
50	**63.2%**
100	62.9%
200	63.1%
400	60.3%
518	63.1%

Table III: Accuracies for our auto-encoder described in secion III-D3, for all tested dimensionalities.

V. Discussion

A. Mel-Spectrogram

The first remark that one can make is the relatively low accuracy, with no model surpassing 70% in accuracy. This is largely due to imbalance inherent from the dataset (figure

3). It is also due to the significant label noise, which stems both from the overlaps that exists between genres and human errors in labelling. Apart from that, one of the main challenges we faced was over-fitting. The dropout and weight decay introduced allowed us to reduce it. Reducing the large gradient-based changes we were experiencing as training went on with an LRS allowed us to stabilise this over-fitting and improve the generalization of our models. We were able to significantly reduce the number of trainable parameters of our model, in large part with the *Divide & Conquer* approach we took, by a factor of more than 30 – from $807k$ parameters to $26k$. Our initial intuition seems to hold: Training models on smaller images and then combining them to generate an ensemble prediction yields desirable results. We also argue that a single recurrent unit is enough for this particular task, both for the smaller track spectrograms and the larger 30s spectrograms, which already halves the amount of trainable parameters. We also found that an optimized model with more than 15 times the parameters of the smaller ensemble-based model performs similarly, which validates the initial *Divide & Conquer* approach. It is also interesting to note that, as we remove convolutional layers, the model trained on the larger spectrograms significantly drops its accuracy, which confirms the need of the initial number of convolutional layers to capture the larger 1292×128 spectrogram's features.

Of course, this approach comes with its drawbacks. In particular, it introduces more pre-processing steps, necessary to divide the tracks into smaller chunks, handling the special cases of variable track length, etc.

B. Manual Features

From the baseline results in section IV-C, table II, we can see that the times taken confirm one of our motivations for going this route to begin with: Lower dimensions do indeed lead to less time taken for training and inference. Furthermore, the achieved scores show us that one can reduce the dimensions to at least 200 – even with such naive methods – without losing significant performance.

Though for our model (see section III-D3 for a description, section IV-C table III for the results), not all dimensionalities lead to results as good as their counterparts in the aforementioned baselines, we can see that the performance does not drop nearly as much when moving into very small dimensionalities. This confirms our hope that intelligent dimensionality reduction allows for much more aggressive compression without loss of accuracy.

Another advantage of our model is the existence of a useful latent space. Through our dual-loss setup, we force the latent space to contain only those features that are most relevant for classification – in other words, those that help differentiate between categories. This effectively means that the structure of the latent space can give us insights into what makes categories different – and also what makes them similar. To this end, we visualized the latent space of our trained auto-encoder with a dimensionality of 50, using the *scikit-learn* implementation of *t-SNE*[4]. The result can be seen in figure 2. This view already allows for some insights.

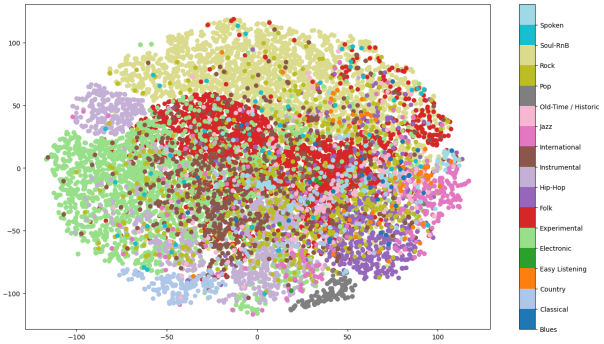

Figure 2: The latent space of our deep auto-encoder after training for both reconstruction and classification. Visualization was done using t-SNE to reduce the number of dimensions from 50 to 2.

For example, it is logical that Rock and Electric music would be largely separate, but that genres like Hip Hop and Experimental would be somewhere between them. We do not go into a full analysis of the latent space here, but careful study of the visualization gives clear reason to believe that the latent space we were able to find is somewhat meaningful and allows for useful interpolation between genres.

On the other hand, one can see that quite a lot of dots do not seem to belong anywhere, and that there is often a considerable overlap between genres. This visually explains the sub-70% accuracy in all our models: The data are apparently very tricky to separate in a meaningful way, hence the subpar performance.

VI. Summary

In this project, we have explored the Deep Learning techniques that exist for the Music Information Retrieval task of Music Genre Recognition. We decided to set the goal of the project to the attempt to reducing the number of parameters, while maintaining an equivalent genre classification performance. We set-out a *Divide and Conquer* approach, where we assessed the behaviour of an ensemble voting over the set of predictions from smaller models of the shorter samples of a track. We conclude that this approach can maintain a comparable accuracy while reducing the number of trainable parameters by a factor of 5 to 30. This comes with the drawback of having additional pre-processing and post-processing steps. On the other hand, – using prediction model based on manually extracted features –, we were able to reduce the number of parameters from 518 to 50 using the latent space representation of the features obtained from a deep auto-encoder; reducing in the meantime the computing resources required.

[4]https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

REFERENCES

[1] M. Schedl, E. Gómez, and J. Urbano, "Music information retrieval: Recent developments and applications," 2014.

[2] B. L. Sturm, "A survey of evaluation in music genre recognition," *Adaptive Multimedia Retrieval*, 2012.

[3] Y. Yaslan and Z. Cataltepe, "Audio music genre classification using different classifiers and feature selection methods," 2006.

[4] "Mel frequency cepstral coefficients for music modeling," *International Symposium on Music Information Retrieval*, 2000.

[5] T. Feng, "Deep learning for music genre classification," 2014.

[6] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks," 2010.

[7] K. Choi, G. Fazekas, and M. Sandler, "Automatic taggine using deep convolutional neural networks," 2016.

[8] P. Chiliguano and G. Fazekas, "Hybrid music recommender using content-based and social information," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[9] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," *World Journal of Education*, 2014.

[10] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," 2016.

[11] M. Hilsdorf, "Music genre classification using a divide conquer crnn," *Medium*, 2022.

[12] A.-K. Al-Tamimi, M. Salem, and A. Al-Alami, "On the use of feature selection for music genre classification," 11 2020, pp. 1–6.

[13] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[14] W. Zhang, W. Lei, X. Xu, and X. Xing, "Improved music genre classification with convolutional neural networks," 2014.

[15] K. Benzi, M. Defferrard, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," *CoRR*, vol. abs/1612.01840, 2016. [Online]. Available: http://arxiv.org/abs/1612.01840

[16] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenbergk, and O. Nieto, "librosa: Audio and music signal analysis in python." 2015.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 3461–3466.

[19] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[20] J. Jakubik, "Evaluation of gated recurrent neural networks in music classification tasks," in *Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology – ISAT 2017*, 2017.

APPENDIX

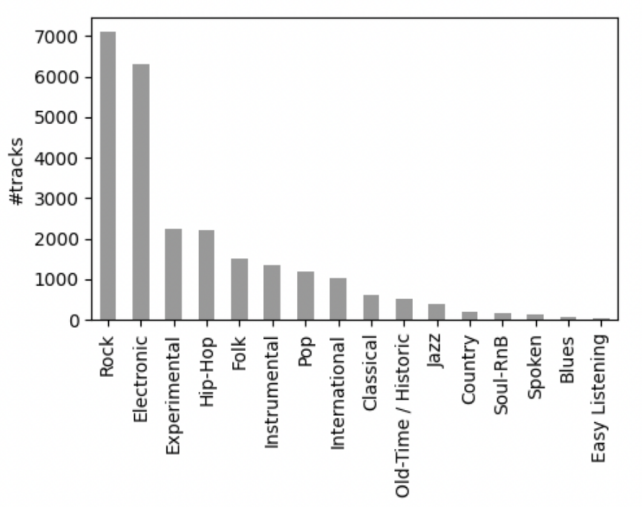

Figure 3: Class Imbalance within FMA

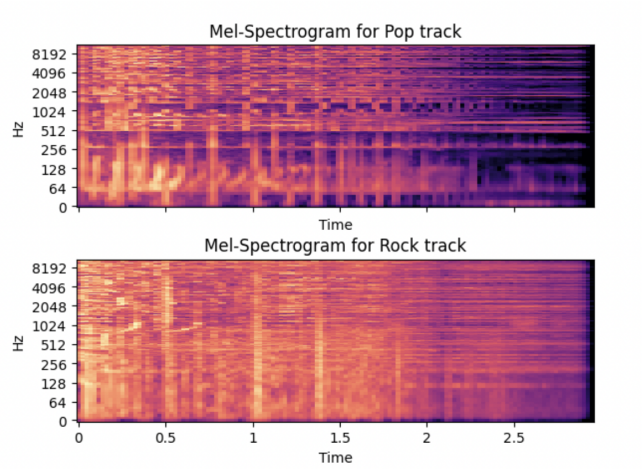

Figure 4: Mel-Spectrogram example for two different genres: Pop and Rock

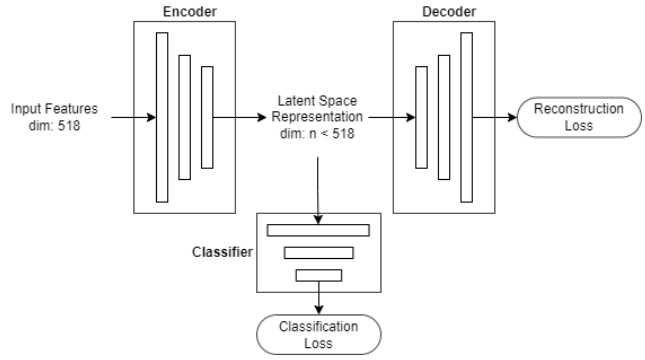

Figure 5: Architecture of our auto-encoder-for-classification model.