Detecting Music Genre Using Extreme Gradient Boosting

Benjamin Murauer
benjamin.murauer@uibk.ac.at
Universität Innsbruck
Innsbruck, Austria

Günther Specht
guenther.specht@uibk.ac.at
Universität Innsbruck
Innsbruck, Austria

ABSTRACT

This paper summarizes our contribution to the CrowdAI music genre classification challenge "Learning to Recognise Musical Genre from Audio on the Web" as part of the WebConference 2018. We utilize different approaches from the field of music analysis to predict the music genre of given mp3 music files, including a convolutional neural network for spectrogram classification, deep neural networks and ensemble methods using various numerical audio features. Our best results were obtained by an extreme gradient boosting classifier.

ACM Reference Format:
Benjamin Murauer and Günther Specht. 2018. Detecting Music Genre Using Extreme Gradient Boosting. In *The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3184558.3191822

1 INTRODUCTION

In the 2018 WebConference task "Learning to Recognise Musical Genre from Audio on the Web", the goal is to predict the music genre of 30 second audio clips automatically. As input for the task, participants are provided with raw mp3 files. These are part of the free music archive [6], which is a collection of over 100,000 music tracks freely available for download. Because the input files are raw audio, multiple steps are required to predict the genre of each track, which are displayed in Figure 1, represented by the gray block descriptions. Firstly, a representation of the tracks has to be found that can be used by classification models, where the type of classifier may determine the type of the features that are to be extracted from the tracks. For example, a convolutional neural network (CNN) may be used together with image features, whereas a random forest classifier requires numerical features.

In our approach, we extract two different kinds of features and a variety of different classifiers to predict the genres of the tracks. The overall workflow is displayed in Figure 1, where details about each step are explained in the respective sections.

The remainder of this paper is structured as follows: In Section 2, related topics and work are discussed. Section 3 describes the task and the dataset in more detail, followed by an explanation of the calculated features in Section 4. All classifiers that have been tested are listed in Section 5, and their performance is discussed in Section 6.

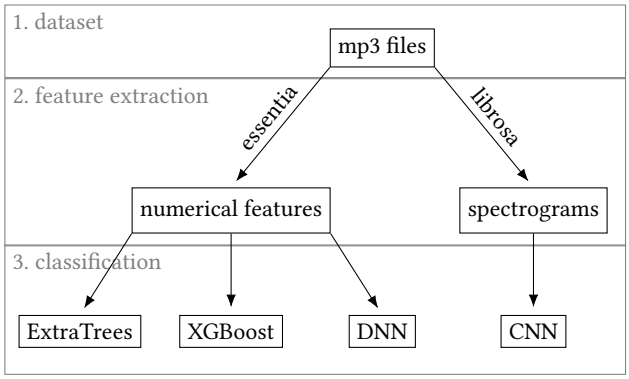

Figure 1: Extracted features and tested classifiers.

2 RELATED WORK

Music genre classification is a well-known objective and many different approaches exist to tackle it. In a similar task held at the MediaEval 2017 workshop [1], the classification labels included subgenres in addition to the main genre. Furthermore, the number of genres a track could have was not limited, resulting in a multi-labeled multi-output classification task. In contrast to the task tackled by this challenge, the organizers of the MediaEval challenge did not provide audio files directly, but rather published precalculated features only. Different solutions proved to be efficient, including a deep neural network (DNN) [10] or hierarchical classification in combination with a voting scheme [13].

While these approaches work with low-level features and computations thereof, other solutions also take music theory into account. Franklin [7] uses long short-term memory (LSTM) cells for extracting high-level features, which can subsequently be used for various purposes. Li et al. [11] have shown that CNNs can be used for extracting features out of the raw audio data, which can then be used for a variety of different tasks.

Other methods for genre prediction use spectrograms (i.e., an image representation of the frequency strengths in a track) in combination with CNNs, transforming the task into an image classification problem [9].

Finally, combinations of CNN and recurrent neural network (RNN) models show improvements over the use of either solution separately. Chen and Wang [2] utilize three different CNNs for different aspects of a spectrogram to calculate high-level descriptors, which are subsequently fed into a LSTM-layer. Costa et al. [5] use a CNN along with a SVM on hand-selected features from the spectrogram image. They then combine these image predictions with the outcome of a SVM trained on acoustical features by fusing the results of both areas with different operations.

WWW'18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3191822

genre	# of songs
Rock	7,103
Electronic	6,314
Experimental	2,251
Hip-Hop	2,201
Folk	1,519
Instrumental	1,350
Pop	1,186
International	1,018
Classical	619
Old-Time / Historic	510
Jazz	384
Country	178
Soul-RnB	154
Spoken	118
Blues	74
Easy Listening	21
total	**25,000**

Table 1: Genre distribution of the traing dataset

In this paper, a collection of different approaches for genre prediction is implemented. We chose to use two ensemble methods, which have been shown to be effective at a similar task [13] and represent an easily calculated baseline. Furthermore, we selected two approaches from current research that have promising results for similar tasks [9, 10]: a DNN which operates on numerical acoustical features and a CNN which works with image representations of the songs.

3 DATASET AND TASK DESCRIPTION

The overall task of predicting the genre of music tracks is split into two parts: in the first phase, contestants have to use a provided script to upload the predicted genres for a provided test set. In the second phase, contestants have to upload a docker image that includes the model of the submitted solution, which is then used to predict genres of a previously unknown second test set. The provided training and test data represents a subset of the free music archive [6]. The training set features 25,000 mp3 formatted audio files, along with meta information containing their true genre. While each of the tracks is 30 seconds long (except for a few broken files), their genres range over both a wide variety of different classes as well as a highly unbalanced distribution, as is depicted in Table 1.

The test set consists of 35,000 equally formatted, but unlabeled mp3 files. For each of these tracks, the respective genre is to be predicted. Contestants are not obligated to provide a hard classification, but rather are allowed to supply probabilities for each genre (e.g., p(Rock)=0.9, p(Electronic)=0.06, p(Hip-Hop)=0.04). To measure how well the predicted genres match the ground truth, two different metrics are predefined by the challenge organizers:

Firstly, the *mean log loss* score (L) was used for primary ranking, which is computed as follows:

$$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_{nc}\ln(p_{nc})$$

where N is the number of samples, C is the number of distinct classes (i.e., genres), y_{nc} is a binary label stating whether the n^{th} sample belongs to class c (i.e., y_{nc} denotes the correct label) and p_{nc} is the probability provided by the submitted solution that the n^{th} sample belongs to class c. As L is a loss measure, lower values mean better predictions.

Secondly, the *mean* F_1 score (F_1^m) is only used for breaking ties within ranks of the same L score. It is defined as

$$F_1^m = \frac{1}{C}\sum_{c=1}^{C} F_1^c$$

where F_1^c denotes the F_1 score (harmonic mean of precision and recall) for a particular class c. However, due to the high accuracy of the grading and the continuous nature of the L metric, it is very unlikely that two solutions will tie and the F_1^m-metric has to be used.

Since the participants will only have to include their best model in the docker image for phase 2 of the challenge, we focussed on exploring different approaches for the first phase. Once the second phase starts, we will optimize our best approach only.

4 FEATURE EXTRACTION

In order to predict the genre of the tracks, we first have to extract features from the raw mp3 files, which can then be fed into various classification models. As is depicted in Figure 1, where the feature extraction is displayed as step two, we use several different classifiers, which require a different input representation of the songs. Therefore, we extract two different sets of features from the audio files: a numerical acoustic feature set, which was extracted using the *essentia* library[1], and image representations of tracks, which were created using *librosa*[2]. In the remainder of this paper, we refer to numerical features as the values extracted by essentia (cf. Table 2), in contrast to the image features extracted by librosa.

Firstly, we extracted a numerical feature set by using the essentia framework for audio analysis. Essentia features a standalone binary application for handling a wide variety of different audio formats, which was chosen for an easier configuration of the docker image, which is required for the second part of the challenge task. Table 2 displays a subset of the features that were extracted using essentia. These range from low-level spectral energy bands to high-level constructed features like danceability (how well can someone dance to this track?).

Several of these features (i.e., the key or scale of a track) are categorical rather than numerical. To use them in a wider variety of different classifying models, they were transformed to a one-hot encoding beforehand. The feature category *rhythm beats position* yields a position for every beat detected by essentia. Since the amount of beats obviously differs between tracks, this leads to a divergence in the amount of features per track. For this reason, all entries for this feature category were discarded. Note that the average distance between those beats is still incorporated in the feature set as *rhythm bpm*.

By default, essentia tries to extract meta information from tracks, including the tracks' artist, album or its genre. As these fields don't

[1] http://essentia.upf.edu
[2] https://librosa.github.io

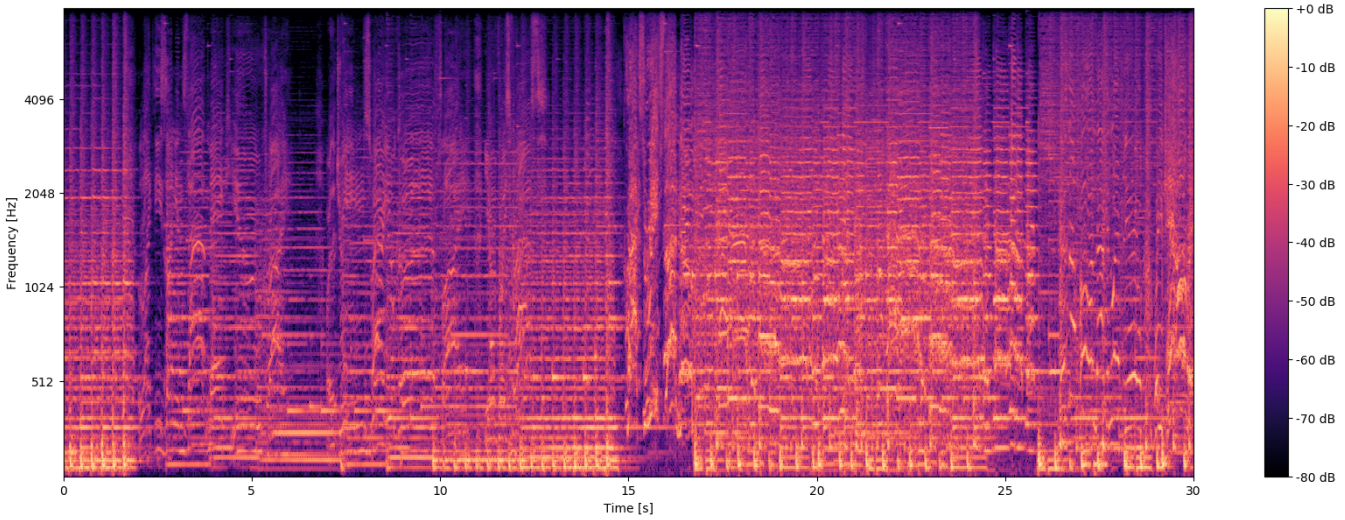

Figure 2: Example of a spectrogram extracted from the test set. Lighter pixels denote a more powerful frequency (y-axis) at the respective time (x-axis)

feature name	exemplary value
low level average loudness	0.938
low level melbands skewness mean	2.246
low level spectral flux median	0.112
rhythm bpm	83.583
...	...
danceability	1.101
tonal key	'E'
tonal chord	'major'

Table 2: Exemplary audio features extracted with essentia. Some of the features are categorical (e.g., tonal key), requiring one-hot encoding for some classification models. All feature values were z-normalized.

represent an acoustic feature and are unavailable for testing, they were removed from the training set.

After these feature selection steps, the amount of features used for classification was reduced from 2,717 to 2,677. Before feeding them into the respective models, all values were normalized to zero mean and 1.0 standard deviation.

Secondly, for extracting image representations, we used the methodology proposed in [9] and calculated mel spectrogram images for each track, which have been shown to be effective in the task of predicting genres [4, 12]. An example for such a spectrogram can be seen in Figure 2. The key idea for this method is that different music genres feature different patterns in the distribution and occurrences of specific frequency ranges, which are displayed in the image. Thereby, the raw frequency is normalized to the mel scale, which more accurately represents a listeners perceived frequency [15]. All tracks from the training and testing set were converted to $500 \times 1{,}500$ pixel images by using the respective functions from the librosa library. Although the original images (i.e., Figure 2) feature a color mapping of the output intensities, the CNN model only uses grayscale pixels in order to save memory. This does not reduce the information stored in the image, as the mapping is linear and merely for producing optically pleasing output for humans.

5 CLASSIFICATION MODELS

To predict the genre of the provided tracks, we rely on machine learning models. Thereby, several different classifiers on the computed features. These are depicted as step three in Figure 1. We divide our approaches into two types, depending on which features extracted in the previous step are used for the respective model.

5.1 Numerical Feature Models

For the numerical feature set, we utilize three different classifying models. We first tested two ensemble classifiers with help of the scikit[3] library:

(1) ExtraTrees is a variant of the random forest classifier and uses extreme random trees for classification [8]. We included it as a reliable baseline approach for comparing other models.
(2) The XGBoost classifier uses extreme gradient boosting [3], which has been shown to be effective in a wide variety of tasks, ranging from recommending jobs [14] to assisting neural networks by weighting feature importances [16]. In addition to its good performance, we chose XGBoost for its versatility and simplicity for parallelization.

We utilized a grid search approach with 5-fold cross validation to tune the parameters for each of the classifiers. Due to time limitations, not all possible parameters were included in the grid search process, which was limited to the amount of trees used (n_estimators), the amount of features used (which resulted in best performance if all features were used) and, in case of XGBoost, the maximal depth of a tree.

[3] http://scikit-learn.org/

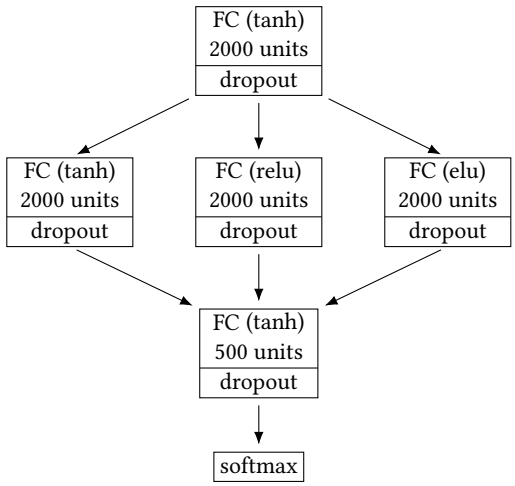

Figure 3: Deep neural network architecture. FC denotes a fully connected layer with the activation function stated in parentheses.

parameter	value
input dimension	2,677
dropout probability	0.5
activation function (input)	tanh
activation function (layer 2)	tanh, relu, elu
activation function (layer 3)	tanh
initializer	He
optimizer	adam
batch size	50

Table 3: Best parameters found for the deep neural network.

In addition to these ensemble methods, we constructed a deep neural network along the lines of the winning solution of the MediaEval 2017 challenge [10]. The architecture of the network is displayed in Figure 3. At every dense layer, a random dropout (with $p = 0.5$) and batch normalization was performed to prevent overfitting. In the second layer, three different activation functions are used to implicitly represent the internal features as good as possible. The next dense layer uses the hyperbolic tangent activation function, as this provided the best results in our experiments. Finally, the last layer uses softmax to calculate the probabilities for each genre. The best performing parameters are listed in Table 3. Because the performance of the DNN was substantially lower than the ensemble approaches, we did not perform an extensive parameter search on this model. Instead, we focussed on increasing the other, more promising solutions.

5.2 Image Feature Model

For the second set of features, where the tracks are represented as images, the classification approach suggested by [9] was used to construct a CNN, which was trained on the spectrograms of the tracks. The architecture of this network is displayed in Figure 4, whereas the detailed parameters that were used are listed in Table 4. The full size of the previously extracted images (i.e., 500 × 1,500 pixels) could not be used due to GPU memory limitations on our computers (GTX1060, 6GB VRAM). Instead, the images had to be downsized to 400 × 1,200 pixels. Instead of using a larger kernel for the convolution operations, we stacked two convolutional layers using smaller kernel sizes (of 3 × 3 pixels) at each convolution step for better memory efficiency. After every pooling layer, a random dropout (with $p = 0.5$) was introduced for regularization. For layout reasons, these layers are not displayed in the diagram.

parameter	value
input dimensions	400 × 1,200 pix., 1 channel
kernel size	3 × 3 pixels
number of filter maps	4
max pooling size	2
batch size	25
dropout probability	0.5
activation function (conv.)	relu
activation function (dense)	tanh
fully connected cells	50
optimizer	adam
initializer	glorot uniform
padding	same

Table 4: Best parameters found for the convolutional network.

classifier	parameters	L	F_1^m
XGBoost	n_estimators=1,000, max_depth=3	**0.82**	0.74
XGBoost	n_estimators=3,000, max_depth=5	0.85	**0.78**
ExtraTrees	n_estimators=1,000	0.92	0.74
ExtraTrees	n_estimators=2,000	0.92	0.74
ExtraTrees	n_estimators=2,000, balanced weights	0.96	0.73
CNN	*	1.65	0.48
DNN	*	1.44	0.77

Table 5: Results and parameters of tested models. * Parameters of DNN and CNN classifiers are listed in Tables 3 and 4, respectively.

6 RESULTS

The results of all classifiers can be seen in Table 5. From all tested models, only ExtraTrees features an automatic balancing of the sample weights according to the class imbalances. However, as is listed in Table 5, this optimization technique yielded a slightly higher loss. For all other classifiers, no explicit measures were taken for tackling the class imbalance. It can be seen that the more traditional ensemble approaches outperform the neural networks, with XGBoost achieving the lowest loss of $L = 0.82$, whereas the CNN performs worst with $L = 1.65$. This result conflicts with the current state of research, where many top tier approaches use neural networks for similar tasks [9, 10].

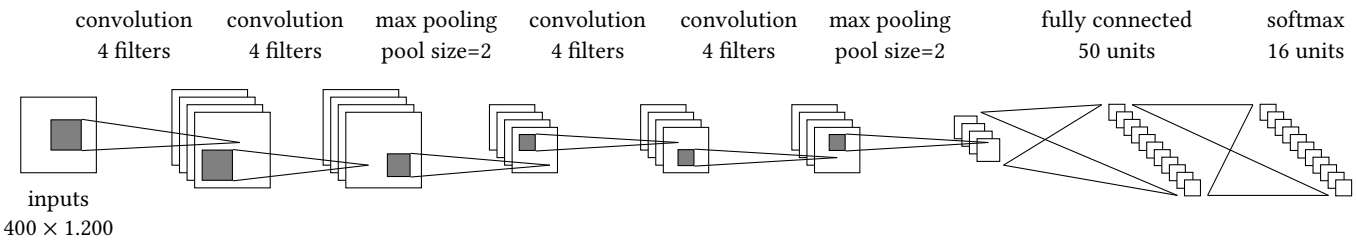

Figure 4: Convolutional neural network architecture. For layout reasons, dropout layers are omitted in this diagram.

At this point, the poor performance of the DNN model may have resulted from various different factors. Given the limited timeframe, we were not able to analyze and identify which network design choices were most significant for the given problem. Interestingly, the F_1^m-score of the DNN approach was comparable to the ensemble solutions.

As of the CNN, we presume that a more accurate model could have been built with more GPU memory. Although a better performing model may have been found with the resources at hand, we were restricted in exploring different network layouts due to hardware limitations. In detail, the memory limited the following parameters to be increased (cf. Table 4):

- number of dense units after the convolution (50)
- number of filters used for each convolution (4)
- batch size (25)
- input dimension (400 × 1,200 pixels)
- number of convolutional layers

As each of these parameters potentially increases the expressiveness of the CNN model (for example, [9] and [4] use 5 convolutional layers each), we assume that testing larger values could have yielded better predictions. Especially the combination of fewer, smaller layers and decreased image size is a possible explanation why the performance of the CNN model is behind other approaches like XGBoost.

7 CONCLUSION

In this paper, we used different types of classifiers to predict the genre of unlabeled music tracks. We extracted two different sets of features, yielding a numerical and a graphical representation of each track. These are used in combination with various models that have been effective for similar problems. For the numerical features, we used ensemble methods (XGBoost, ExtraTrees) as well as a deep neural network for classification. The graphical features were fed into a CNN. Our best results were obtained by the XGBoost classifier on the numerical feature set, yielding a mean log loss of $L = 0.82$, compared to 0.92 for the ExtraTrees approach and 1.44 and 1.65 for the DNN and CNN models, respectively. Many promising approaches, especially more elaborate neural networks, could not be implemented or optimized due to GPU memory limitations.

REFERENCES

[1] Dmitry Bogdanov, Alastair Porter, Julián Urbano, and Hendrik Schreiber. The MediaEval 2017 AcousticBrainz Genre Task: Content-based Music Genre Recognition from Multiple Sources. In *Working Notes Proceedings of the MediaEval 2017 Workshop*. CEUR-WS.org, 2017.

[2] Ning Chen and Shijun Wang. High-level music descriptor extraction algorithm based on combination of multi-channel cnns and lstm. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR'2017)*, pages 509–514, 2017.

[3] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

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

[5] Yandre M.G. Costa, Luiz S. Oliveira, and Carlos N. Silla. An evaluation of convolutional neural networks for music classification using spectrograms. *Applied Soft Computing*, 52:28 – 38, 2017.

[6] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A Dataset for Music Analysis. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR'2017)*, 2017.

[7] Judy A. Franklin. Recurrent neural networks for music computation. *INFORMS Journal on Computing*, 18(3):321–338, 2006.

[8] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[9] Grzegorz Gwardys and Daniel Grzywczak. Deep Image Features in Music Information Retrieval. *International Journal of Electronics and Telecommunications*, 60(4):321–326, 2014.

[10] Khaled Koutini, Alina Imenina, Matthias Dorfer, Alexander Rudolf Gruber, and Markus Schedl. MediaEval 2017 AcousticBrainz Genre Task: Multilayer Perceptron Approach. In *Working Notes Proceedings of the MediaEval 2017 Workshop*. CEUR-WS.org, 2017.

[11] Tom LH. Li, Antoni B. Chan, and Andy HW. Chun. Automatic Musical Pattern Feature Extraction Using Convolutinoal Neural Network. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2010.

[12] Thomas Lidy and Alexander Schindler. Parallel convolutional neural networks for music genre and mood classification. *MIREX 2016*, 2016.

[13] Benjamin Murauer, Maximilian Mayerl, Michael Tschuggnall, Eva Zangerle, Martin Pichl, and Günther Specht. Hierarchical Multilabel Classification and Voting for Genre Classification. In *Working Notes Proceedings of the MediaEval 2017 Workshop*. CEUR-WS.org, 2017.

[14] Andrzej Pacuk, Piotr Sankowski, Karol Węgrzycki, Adam Witkowski, and Piotr Wygocki. Recsys challenge 2016: Job recommendations based on preselection of offers and gradient boosting. In *Proceedings of the Recommender Systems Challenge*, pages 10:1–10:4, 2016.

[15] Stanley Smith Stevens, John Volkmann, and B. Edwin. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

[16] Huiting Zheng, Jiabin Yuan, and Long Chen. Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation. *Energies*, 10(8), 2017.