Comparing Meta-Classifiers for Automatic Music Genre Classification

Vitor Yudi Shinohara [1]*, Juliano Henrique Foleiss[1] , Tiago Fernandes Tavares[1]

[1] School of Electrical and Computer Engineering - University of Campinas - Brazil
Albert Einstein, 400 – 13083-852, Campinas, SP

vitorys@dca.fee.unicamp.br, julianofoleiss@utfpr.edu.br, tavares@dca.fee.unicamp.br

Abstract. *Automatic music genre classification is the problem of associating mutually-exclusive labels to audio tracks. This process fosters the organization of collections and facilitates searching and marketing music. One approach for automatic music genre classification is to use diverse vector representations for each track, and then classify them individually. After that, a majority voting system can be used to infer a single label to the whole track. In this work, we evaluated the impact of changing the majority voting system to a meta-classifier. The classification results with the meta-classifier showed statistically significant improvements when related to the majority-voting classifier. This indicates that the higher-level information used by the meta-classifier might be relevant for automatic music genre classification.*

1 Introduction

Music genres are categories that group songs with same characteristics, such used instruments, rhythm and music's harmony [1]. It can be used to organize music track collections [2]. Automatic Music Genre Classification (AMGC) is a Music Information Retrieval task that aims at facilitating the labeling of tracks according to their genre [1].

AMGC relies on representing music tracks in a vector space using sound-related features. Some approaches use a single vector to represent each track, whereas others use multiple vectors for this representation. A single-vector representation (SVR) is more compact, but can result in loss of information due to long-term summarization [3]. Conversely, the multiple-vector representation (MVR) depends on an additional step for combining the information derived from each of the vectors [4]. Each vector in the MVR representation represents different sounds from the track, leading to a richer representation.

A possible approach for combining information from MVR is using voting mechanisms, in particular, majority voting [4, 5]. This method relies on classifying each frame individually and then selecting the most frequent label as the one associated to the track. It relies on the idea that classification errors are less frequent than correct classifications, hence having multiple attempts for each track reduces the probability of an overall error.

In this paper, we explore diverse methods for combining texture classifications into a final prediction per track. For such, we use the outputs of a texture genre classifier as the input for a meta-classification stage. This relies on the hypothesis that some classification errors are typical in particular genres, thus these errors can be exploited to improve classification results.

We evaluated three different approaches for combining the classifications in the meta-classification stage. We considered the case where the output for each texture is just the predicted class, not the estimated probability for each class. The baseline is the widely used majority voting [6, 7, 8, 9, 10]. We evaluated meta-classifiers based on two different representations. In the first one, the classification histogram is used as a feature vector, which is yielded to the meta-classifier. Second, we used the sequence of texture class predictions as inputs to time-series classifiers.

The approaches were evaluated in four different datasets. Our results indicate that majority voting is an effective technique for datasets containing full-length popular music tracks, and time-series classification is more accurate when textures are typically more uniform throughout the track. This suggests that the changes in musical textures throughout a track can be relevant for genre classification, but they can be hard to model using general-purpose tools in heterogeneous tracks.

This paper is structured as follows: In Section 2 we introduce the meta-classifier architecture, features and data sets used in the evaluation. In Section 3, meta-classifier results are compared with majority voting results. In Section 4, final considerations are presented.

2 Method

The evaluation system is composed of two stages: texture classification and meta-classification. The first stage outputs a class prediction for each texture of an input music track. In the second stage, the textures of a music track are used to build a track representation that is input into a meta-classifier, which yields a final classification for the track.

2.1 Music Texture Classifier System

The Music Texture Classifier System (MTCS) outputs a class prediction for each texture of an input music track. A texture is a feature vector aggregated over a sequence of feature vectors calculated over audio frames. Textures aim to encode audio content of a relatively long (typically 1s to 5s) audio segment, which is useful for genre classification [1]. Figure 1 shows the MTCS architecture. Given a music track sampled at 44Khz, a 2048-sample Short-Time Fourier Transform (STFT) is calculated, with 50% overlap. This yields 23ms frames. Then, a set of handcrafted features are calculated for each frame. This feature set consists of the following features: Spectral Cen-

*Supported by CAPES.

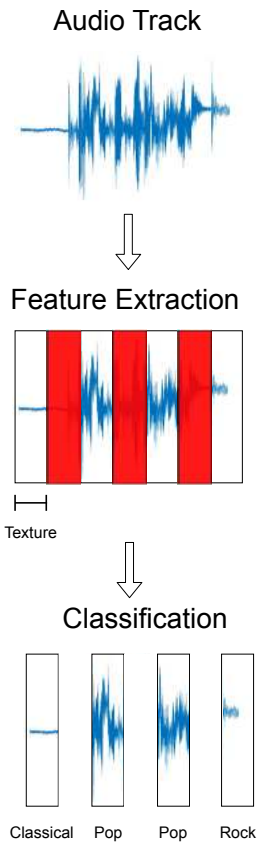


Figure 1: Music Texture Classifier System (MTCS) architecture. It receives an audio file as input and yields a sequence of label predictions related to each segment of the track.

troid, Spectral Rolloff, Spectral Flux, Energy, Zero Crossing Rate [1], Spectral Flatness [11], and the first 20 MFCC coefficients [12]. The feature vector also consists of the first and second-order derivatives of each feature. Thus, each frame-level feature vector has 78 features.

Textures are calculated using the mean and variance of each feature in every 10 low-level frames, resulting in a 10x downsample. This yields a sequence of 156-dimensional feature vectors.

However, the total number of textures in the training set can become too large. To make training tractable, the texture set for each track is further downsampled by selecting k linearly-spaced textures. Since k is a parameter, it was evaluated as 5, 20 and 40 in our experiments.

Along with the votes for each texture of a track, the MTCS also yields a final label for each track. This label is computed by a voting procedure, in which the class with the most votes is decided as the track label. These results are used as baseline.

2.2 Meta Classification

This paper explores how the MTCS votes can be combined via meta-classifiers. The votes were organized in two different representation as input for the meta-classifier: vote histograms and sequences of votes. Histograms represents the number of votes for all genres in a specific track. Sequences of votes indicate the vote progression along the track. In other words, histograms disregard the order of the textures, whereas sequences of votes rely on this information.

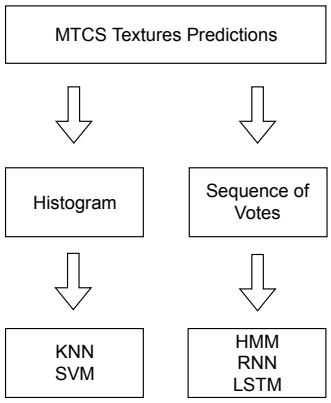


Figure 2: Histogram and sequence of votes classification scheme.

2.2.1 Histogram Classification

The left-hand side of Figure 2 shows how histograms are used in our meta-classification approach. First, texture votes are obtained from the MTCS for every track in the dataset. Then, vote histograms are built by computing the number of votes each class received. These histograms are used to describe music tracks. The true track labels were used as target values. Two classifiers were evaluated as meta-classifiers, The K-Nearest Neighbor (KNN) and the Support Vector Machine (SVM).

The systems were evaluated using K-fold cross-validation. To make it easier to compare to the baseline, the same folds were used to evaluate the histogram meta-classifiers. Thus, the same training sets were used for training the meta-classifiers, while the same testing sets were used for evaluating them.

For hyper-parameter tuning, the training set was randomly split into a training set (80%) and a validation set (20%). A grid-search was used for hyper-parameter tuning. The classifier parameters evaluated are shown in Table 1.

Table 1: Hyper-parameters evaluated with grid-search for histogram classifiers.

Classifier	Hyper-parameter	Range
SVM	*C*	{ 0.001, 0.01, 0.1, 1, 10, 100 }
SVM	*Gamma*	{ 0.001, 0.01, 0.1, 1 }
KNN	*k*	{ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 }

2.2.2 Sequence of Votes Classification

The right-hand side of Figure 2 shows how the sequences of votes are used for meta-classification. The MTCS outputs the votes following the original sequence of the corresponding textures in the track. The sequences are then

used as input to the meta-classifier. Three sequence classifiers were evaluated with the sequences as input. Hidden Markov Models (HMM) [13] can be used as classifiers. A HMM is trained for each target class. Then, test sequences are evaluated, and each HMM predicts the probability it generated the sequence. The target class corresponding to the maximum probability HMM is assigned to the sequence. Grid-search was used to find the best hyper-parameter combination on the validation set. Table 2 presents the hyperparameter evaluation values used in our experiments.

Table 2: HMM Hyper-parameters tuned with grid-search.

Hyper-Param	Tested Values
Number of Hidden States	{3, 5, 7, 9}
Covariance Matrix	{full, diag}
Number of Iterations	{50, 100, 150}

We also evaluated Recurrent Neural Networks as meta-classifiers. Table 3 presents the architecture used in our experiments, along with the evaluated parameter values. Two types of recurrent cells were evaluated: simple Recurrent Neural Network (RNN) and Long ShortTerm Memory (LSTM) [14]. The Long Short Term Memory mitigates the vanishing gradient problem that occurs in traditional RNNs. This problem is known to get worse as the sequences get longer.

Table 3: Neural network architecture used in the experiments.

Layer	Type	Activation Function	Neurons
1	Dense	Linear	1
2	RNN / LSTM	tanh	{20, 30, 40, 50}
3	Dense	ReLu	{5, 10, 15}
4	Dense	Softmax	{9, 10, 13}

2.3 Datasets

The datasets used in the experiments are presented in Table 4. These datasets are widely used by the MIR community and are publicly available. These datasets vary greatly in terms of music content, label balancing, number of tracks and track length.

Table 4: Description of evaluation datasets.

Data set	# Tracks	# Genres	Balanced	Track Len.	Folds
GTZAN	1000	10	Yes	30 s	10
LMD	1300	10	Yes	Full	3
HOMBURG	1886	9	No	10 s	10
EXBALLROOM	4180	13	No	30 s	10

The GTZAN dataset [1] is one of the most widely used datasets in genre recognition research [15]. It consists of 10 western genres, which greatly vary with respect to spectral patterns. The folds were created randomly. The HOMBURG dataset [16] presents a challenge for systems based on texture classification, since the tracks are only 10s long. The Extended Ballroom dataset (EXBALLROOM) [17] is also challenging for texture classification systems. Because this dataset is made of ballroom dances, there are subsets of genres that use the same instrumentation. Thus, time-related features, such as rhythm and tempo descriptions, are needed in order to achieve good results. An artist filter [18] was applied to EXBALLROOM during fold splitting. A subset of the Latin Music Database (LMD) [19] was also used in the evaluation. This is the only dataset evaluated that consists of full-length tracks. A subset of the original dataset was used for artist filtering, since some genres were largely represented by only a few artists.

In the next section we present the evaluation results.

3 Results

In this section we present the classification accuracy for all the meta-classification approaches presented in this paper. The results are the average and standard deviation across all folds. The number of folds varies depending on the dataset and are shown in Table 4. Statistical significance was evaluated by the Student's T-test. Statistical significance was evaluated for all meta-classification approaches when compared to the Majority Vote baseline. A threshold of 5% was considered for rejecting the null-hypothesis.

Table 5 shows the best results for each classifier in the four datasets evaluated. Statistically significant results are shown in bold.

The results from KNN and SVM were similar. For all results, the difference is not statistically significant. KNN is known to be able to perform well in low-dimensional data. As the largest feature vector had 40 features, the dimensionality did not have a big impact on the KNN results. Furthermore, KNN has a lower training computational cost. Thus, in the evaluated datasets, KNN offers a superior cost/benefit ratio. However, only the SVM was statistically superior than the majority voting baseline.

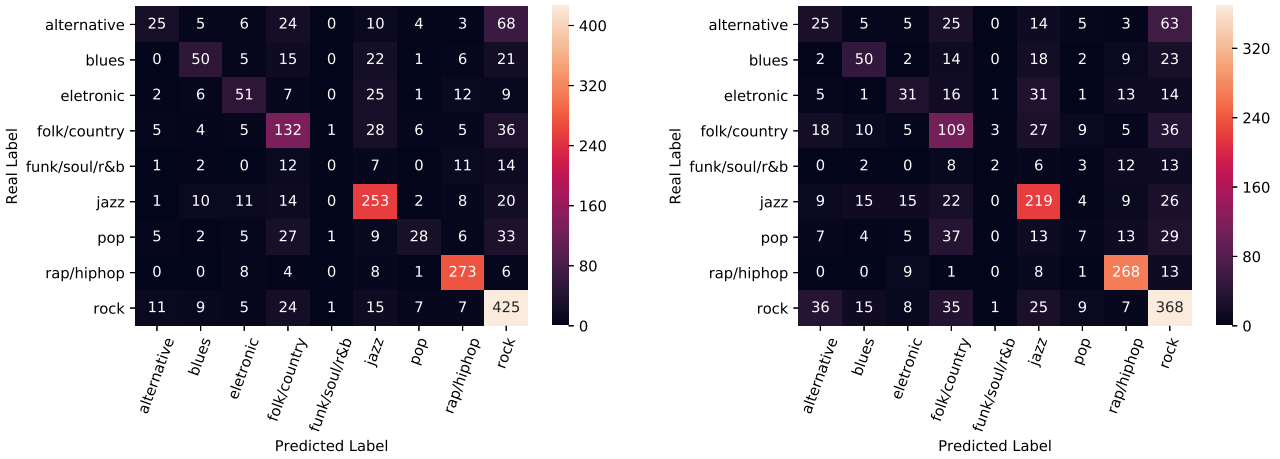
Both neural network results were not statistically superior to the histogram results. This suggests that for the datasets evaluated, the sequence of the votes is not key for performance improvement. Similarly to the SVM, both the RNN and LSTM networks performed better than the baseline.

The confusion matrix for the RNN metta-classifier and the HOMBURG dataset is shown in Figure 3. This was the best average result obtained overall for this dataset ($65\% \pm 0.08$). Figure 3 shows the confusion matrix for the majority voting baseline.

Overall, the results of the histogram-based approaches were similar compared to the majority vote. The majority voting structure is embedded in cases where the histogram was correctly classified by majority vote. When presenting a histogram whose majority vote is correct, the classifier tends to associate the behavior of the majority vote. Therefore, the majority vote seems to represent a lower limit for histogram classification in the evaluated datasets.

Table 5: Best results for all meta-classifiers.

	Baseline	Proposed Method				
		Histograms		Sequence of Votes		
	Majority Vote	**KNN**	**SVM**	**HMM**	**RNN**	**LSTM**
GTZAN	0.79 ± 0.06	0.75 ± 0.16	0.76 ± 0.15	0.53 ± 0.04	0.73 ± 0.14	0.79 ± 0.16
LMD	0.82 ± 0.02	0.81 ± 0.03	0.82 ± 0.02	0.69 ± 0.01	0.72 ± 0.07	0.75 ± 0.07
HOMBURG	0.54 ± 0.03	0.58 ± 0.05	$\mathbf{0.59 \pm 0.04}$	0.53 ± 0.02	$\mathbf{0.65 \pm 0.08}$	$\mathbf{0.64 \pm 0.07}$
EXBALLROOM	0.75 ± 0.02	0.77 ± 0.05	0.77 ± 0.05	0.62 ± 0.05	0.76 ± 0.04	0.78 ± 0.04



(a) RNN meta-classifier.

(b) Majority Voting baseline.

Figure 3: Confusion matrices for the RNN meta-classifier and the baseline systems in the HOMBURG dataset.

4 Conclusion

Various research on Automatic Music Genre Classification use the multiple-vector representation to describe tracks. When only the votes, no probabilities, are available for each texture, a final track classification is decided by majority voting. This paper presented two alternative approaches for combining texture votes into genre predictions. The first method evaluated builds a vote histogram. This histogram is used to represent the track for the meta-classifier, maps histograms into final genre decisions. The second method relies on the sequences of votes for each track. The sequences are then input into sequence classifiers, which map sequences of votes into genres.

The histogram classifiers tends to have a better performance when music textures are more uniform through time. In contrast, the meta-classifiers based on sequences of votes obtain better results on data sets in which musical textures are more heterogeneous. This suggests that LSTM and RNN architectures were effective in modelling the short-term (close to 10s) sound changes that characterize textures, but were unable to derive differences related to musical structure.

Therefore, recurrent neural networks are effective model dependencies on short, few-seconds scale, whereas long-term dependencies should be investigated using other models. The exploration of model behaviors for both of these time scales is an interesting venue for future work.

In conclusion, the source code from the proposed meta-classifiers is available at a public GitHub repository[1]. Instructions for experiment execution and datasets used are also included.

5 Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior - Brasil (CAPES) - Finance Code 001.

References

[1] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002.

[2] CBS Interactive. Last.fm, 2002. Accessed: 2019-05-01.

[3] J. Lee and J. Nam. Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. *IEEE Signal Processing Letters*, 24(8):1208–1212, Aug 2017.

[4] Tiago Fernandes Tavares and Juliano Foleiss. Automatic music genre classification in small and ethnic datasets. In *13th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, sep 2017.

[5] Mathieu Lagrange, Grégoire Lafay, Boris Defreville, and Jean-Julien Aucouturier. The bag-of-frames approach: a not so sufficient model for urban soundscapes. *Journal of the Acoustical Society of America*, 138(5):487–492, October 2015.

[1] https://github.com/vitorys/MusicGenreMetaClassifier

[6] Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. Unsupervised learning of sparse features for scalable audio classification. In *12th Proceedings of the International Conference on Music Information Retrieval*, 2011.

[7] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *12th Proceedings of the International Conference on Music Information Retrieval*, 2011.

[8] Jan Wülfing and Martin A Riedmiller. Unsupervised learning of local features for music classification. In *13th Proceedings of the International Conference on Music Information Retrieval*, 2012.

[9] Il-Young Jeong and Kyogu Lee. Learning temporal features using a deep neural network and its application to music genre classification. In *17th Proceedings of the International Conference on Music Information Retrieval*, 2016.

[10] 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.

[11] S. Dubnov. Generalization of spectral flatness measure for non-gaussian linear processes. *IEEE Signal Processing Letters*, 11(8):698–701, Aug 2004.

[12] M. Hunt, M. Lennig, and P. Mermelstein. Experiments in syllable-based recognition of continuous speech. In *ICASSP '80. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 880–883, April 1980.

[13] Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[15] Bob L Sturm. A survey of evaluation in music genre recognition. In *International Workshop on Adaptive Multimedia Retrieval*, pages 29–66. Springer, 2012.

[16] Helge Homburg, Ingo Mierswa, Bülent Möller, Katharina Morik, and Michael Wurst. A benchmark dataset for audio classification and clustering. pages 528–531, 01 2005.

[17] Helge Homburg, Ingo Mierswa, Bülent Möller, Katharina Morik, and Michael Wurst. A benchmark dataset for audio classification and clustering. pages 528–531, 01 2005.

[18] Elias Pampalk, Arthur Flexer, and Gerhard Widmer. Improvements of audio-based music similarity and genre classificaton. In *Proceedings of the 6th International Conference on Music Information Retrieval*, 2005.

[19] Carlos Silla, Alessandro Koerich, and Celso Kaestner. The latin music database. pages 451–456, 01 2008.