Applied Soft Computing Journal 89 (2020) 106127

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Texture selection for automatic music genre classification

Juliano Henrique Foleis [a,b,*], Tiago Fernandes Tavares [a]

[a] School of Electrical and Computer Engineering, University of Campinas, Campinas, Brazil
[b] Department of Computing, Universidade Tecnológica Federal do Paraná, Campo Mourão, Brazil

ARTICLE INFO

Article history:
Received 6 June 2019
Received in revised form 26 October 2019
Accepted 24 January 2020
Available online 1 February 2020

Keywords:
Music genre classification
Sound texture selection
Music classification
Signal processing
Music information retrieval

ABSTRACT

Music Genre Classification is the problem of associating genre-related labels to digitised music tracks. It has applications in the organisation of commercial and personal music collections. Often, music tracks are described as a set of timbre-inspired sound textures. A subset of the sound textures is often selected to represent the entire track. In this paper, we evaluate the impact of texture selection on automatic music genre classification. Although previous work has selected textures by linear downsampling, no extensive work has been done to evaluate how texture selection benefits music genre classification. We also present a novel texture selector based on K-Means aimed to identify diverse sound textures within each track. Our results show that capturing texture diversity within tracks is important towards improving classification performance. Our results also indicate that our K-Means based texture selector is able to achieve significant improvements over the baseline with fewer textures per track than the other texture selectors evaluated. We also show that using multiple texture representations allows further opportunities for feature selection to improve classification performance.

1. Introduction

In this paper we investigate the effect of selecting a subset of music track segments to describe tracks in an automatic music genre classification system. Besides that, we present a selector based on K-Means clustering. We also discuss the importance of capturing diverse segments from each track towards classification performance.

Genre is a descriptive tag commonly associated with music tracks. It relates to the social groups that participate in the process of producing, marketing and consuming particular music pieces or styles. Music genre also relates to the instruments and techniques used for musical composition, performance, and perception, and this reflects on the spectral patterns that are present in its digital audio recordings. Such patterns can be exploited to devise mathematical models for sound perception that allow automatically associating genre tags to digital music, that is, automatic Music Genre Classification (MGC) [1]. This task is also known as Music Genre Recognition (MGR).

Many MGC systems assume that spectral patterns that allow predicting genre are typically a few (1 to 5) seconds long. These patterns are represented by vectors known as textures [1]. Fig. 1 presents a typical texture construction procedure. Such vectors are constructed by aggregating a sequence of frame-level feature vectors with low-order statistics such as mean and variance. There may be some overlap between textures. In this paper we refer to the feature vectors of low-level audio frames as *low-level features*, while their aggregation over time is referred to as *textures*.

Often, music tracks are represented by a collection of textures. This allows a rich description of the underlying genre tags because the collections comprise distinct sound textures that are present in each musical piece. Many MGC approaches use collections of textures and disregard their time-domain ordering. These approaches can be seen as variants of the Bag of Frames (BoF) technique [2]. There are many approaches for combining the textures in classification, including statistical modelling [2], dictionary learning and quantisation [3,4], and using a subset of textures to describe tracks [5–12].

We call Texture Selection the techniques used to select a subset of textures to describe music tracks. One of the concerns related to Texture Selection is that computing costs increase as more textures are selected. We are not aware of any previous research that evaluated the effect of texture selection in music genre classification. In this paper we present such a study. We also propose to a novel texture selection technique in which textures are selected separately for each track via K-Means clustering. Each track is represented by a collection of its centroids. We call this technique KMEANSC. K-Means clustering highlights distinct audio trends which are able to capture timbre variety within tracks, increasing feature space coverage and promoting generalisation, as shown later.

* Corresponding author at: Department of Computing, Universidade Tecnológica Federal do Paraná, Campo Mourão, Brazil.
E-mail address: julianofoleiss@utfpr.edu.br (J.H. Foleis).

https://doi.org/10.1016/j.asoc.2020.106127
1568-4946/

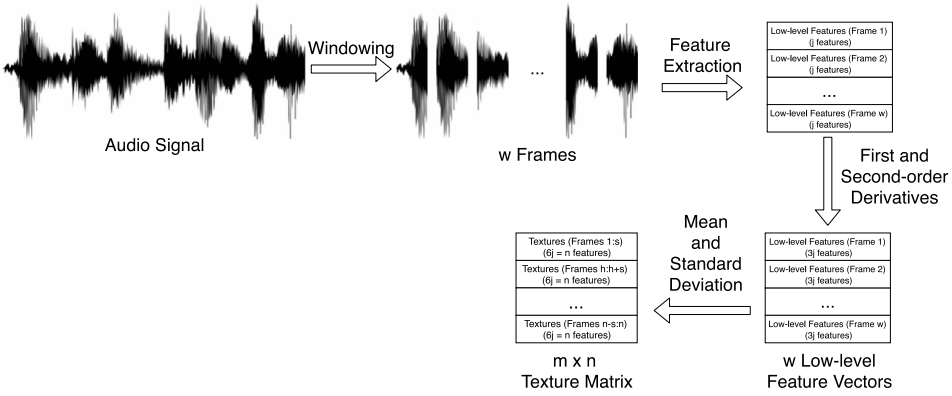


Fig. 1. Typical Texture Construction Procedure. The audio signal is windowed into w overlapping frames. Features are extracted for every frame into j-dimensional feature vectors. First and second order deltas are appended to the feature vectors, resulting in $3j$-dimensional feature vectors. Finally, m textures are constructed by aggregating s consecutive feature vectors with the average and standard deviation, with a hop size of h, resulting in n-dimensional vectors, $n = 6j$. The output is a texture matrix in $R^{m \times n}$.

We compare the results of our texture selection approach to linear downsampling, which is widely used in previous research [5–12]. Our results show that, in the datasets employed in this work, K-Means centroid downsampling leads to higher classification accuracy using less textures than linear downsampling. Also, feature-space analysis indicate that this improvement depends on the presence of texture diversity within the tracks.

We also investigate how texture selection compares with feature selection. Our results show that representations based on multiple textures per track offer a greater opportunity for feature selection to improve classification performance. We also evaluated the effects of texture selection in different feature sets. Handcrafted features, data-driven features and random projections of Mel-spectrograms were evaluated in the experiments. Our results show that the effects of texture selection are replicated in all feature sets we evaluated.

This paper is organised as follows: Section 2 presents related work concerning track representations and texture selection. Section 3 describes our method, which encompasses the description of our proposed KMEANSC texture selector. In this section we also describe the other texture selectors, the feature sets and feature selection algorithms evaluated. In Section 4 we present our experimental setup, describing the experiments we performed, as well as all the parameters needed to replicate our results. In Section 5 we present the datasets used in the evaluation. In Section 6, we present the results of our texture selection experiments. In Section 7 we present a discussion on the results obtained and derive some insight on the effects of texture selection in music genre classification. Finally, in Section 8, we present conclusions and a path for future work.

2. Related work

The objective of texture selection is to describe a music track by highlighting a subset of its textures in such a way that it reduces computing and storage requirements, while improving classification performance. In this section we present previous music track descriptions and texture selection approaches.

There are mainly two music track description approaches: global or bag of frames descriptions. Global descriptor approaches use a single vector to describe an entire track. A common global description approach was first described by Tzanetakis and Cook [1]. In their method all textures of a track are averaged feature-wise into a single texture. We call their approach Full Track Statistics (FTS). Many modern works also use this approach. In [13], the audio signal is decomposed using the undecimated wavelet transform (UWT). Then, each sub-band signal is split into frames. Low-level features are extracted from each frame and are then combined into a single vector per track using various statistical moments. In [14], they classify albums into genres. Each track is described by a CQT spectrogram. For a given album, 15-second patches from every track are averaged into a single patch that represents the entire album.

The main advantage of FTS descriptions is that model training is usually fast, since only a single vector describes the entire track or album. However, it assumes that each track can be fully described by an "average" texture. This simplification fails to leverage the feature space spanned by the entire track, collapsing it into a single point. Over all tracks in a dataset, this leads to a poorer description of the feature space, impacting generalisation.

Bag of frames approaches acknowledges that music track is made up of a set of varied textures. Aucouturier et al. [2] popularised the term Bag of Frames, arguing that it is similar to how the Bag of Words approach in Natural Language Processing represents texts by a global word distribution, disregarding their sequence.

Aucouturier and Pachet [15] introduced a representation in which tracks are described by a Gaussian Mixture Model (GMM) that models the probability of any texture being associated with the track. Pampalk et al. [16] later used this representation for music genre classification. They used a nearest neighbour classifier with a Monte Carlo approximation of a distance metric based on log-likelihoods of the query track GMM, and the GMMs of the tracks in the training set. This approach is very computationally expensive, and is not suitable for large datasets [16].

Dictionary learning is another bag of frames approach. Marques et al. [3] uses K-Means to learn codewords from each track of the training set separately. Then, a track is described by a histogram of centroids closest to the track textures. The main drawback of this approach is that the histograms can become too large as the number of tracks in the training set increases.

A widely used bag of frames approach consists in downsampling the set of textures of a track into a subset. This subset represents the entire track. Texture selection, which is the main focus of our contributions, deals with this approach. A common downsampling approach selects linearly-spaced textures along time, which we call LINSPACE. Yandre et al. [5] uses a Convolutional Neural Network (CNN) to classify tracks into genres. Each track is clipped to the middle 60 s of audio. A STFT is computed for each track, resulting in a 256×800 spectrogram. This spectrogram is then sliced into a set of 50 non-overlapping 256×16 patches, representing 256 frequency bins and 16 audio

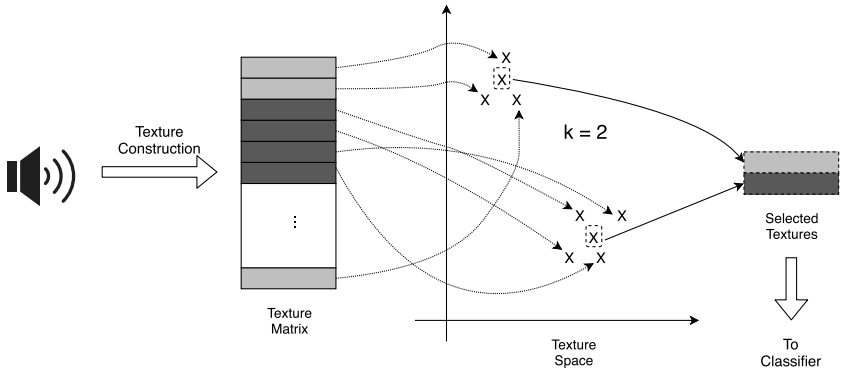


Fig. 2. K-Means Texture Selection. Texture construction is detailed in Fig. 1. K-means clustering identifies acoustic trends in the music track, represented by the resulting cluster centres. The collection of centroids is selected as the track description. k is a parameter that must be optimised according to the dataset.

frames. The patches are the input to the network. Every patch from the same track is given the track label for training. The network outputs the probability of each genre for every patch. A final classification is decided by the sum rule, where the genre probabilities are added for every patch of the same track. The genre with maximum probability is assigned to the track.

Various other works have also used LINSPACE as a texture selection method. Recently, Kim et al. [6] used a neural network that takes raw waveforms as input. For the Music auto-tagging task, tracks are sliced into 10 non-overlapping segments, each with 59,049 samples. The network outputs the class probabilities independently for each segment and a final decision is made by averaging the tag predictions for all segments. Yang and Zhang [7] described music tracks in a similar fashion to [5], but used Mel Spectrograms instead of the STFT. The patch size was 256×128, each covering 3s of the track. A genre is predicted for each patch, and the final track prediction is decided as in [5]. Senac et al. [8] describes tracks either by STFT spectrograms or handcrafted features. For either feature set, the track is sliced into 3s clips with 50% overlap. A final decision is made by majority voting. Further recent works also uses LINSPACE as a texture selection method [9–12].

LINSPACE does not take content into consideration. Rather, it selects linearly-spaced textures along time. While it acknowledges that music tracks are made up of set of varying textures, it does not pursue such variety. Selecting varied textures is left to chance. As with FTS, this approach fails to leverage the feature space spanned by the entire track. However, it encompasses a subset of the possible textures, instead of collapsing it into a single point as FTS.

Another disadvantage of LINSPACE is that the computational costs rise as the number of textures per track increases. This particularly affects training times. Thus, it is important to find a reasonable trade-off between computing cost and system performance. We searched the literature for previous works that discusses issues related to texture selection. We were not able to find any previous research that deals with how texture selection affects classification performance. Particularly we were looking for answers to questions such as:

- How many textures are needed to get a reasonable classification performance?
- Are there texture selection techniques that are able to offer a better trade-off in terms of number of textures and classification performance?
- What does an appropriate set of textures look like in terms of feature space?

In this work we present a study that aims to shed some light into these questions.

To evaluate how the texture selection technique impacts results and the trade-off described above, we propose a texture selection strategy based on K-Means clustering. K-Means finds the acoustic trends in each track separately. Then, the set of centroids is used to describe the track. Intuitively, we expect the textures selected by KMEANSC to represent a greater amount of texture possibilities per genre, spanning more of the feature space and promoting generalisation. We are not aware of any previous works that used K-Means clustering for texture selection. We are also not aware of any other texture selection techniques besides variations of LINSPACE in the context of automatic music genre classification.

3. Method

One of the main concerns presented by previous research using collections of textures to describe music tracks is the amount of computing power needed to train models. Texture selection can be used to reduce the number of textures per track, while preserving classification performance. In this section, we present a method for selecting representative textures of a music track by representing it with K-Means centroids. We compare it to other texture selection models used in the literature such as LINSPACE and FTS. The texture selection mechanisms are evaluated under varied conditions such as the presence of feature selection and changing feature sets.

3.1. K-means texture selection — KMEANSC

K-Means is a well-known clustering algorithm introduced by Macqueen in [17]. It works by iteratively estimating points, known as *centroids*, that characterise different trends in a dataset. These points can then be used to query the dataset for other points following the trend.

As stated before, each audio texture is represented by a vector in a perceptually-inspired space $\mathbb{R}^n$, where n is the number of features that span the space. Thus, a music track can be described by a texture matrix $T \in \mathbb{R}^{m \times n}$ where m is the number of textures in the track and n is the number of features. When K-Means is applied to T, it estimates a centroid matrix $C \in \mathbb{R}^{k \times n}$, where k is the number of centroids. These centroids can be interpreted as vector representations of the acoustic trends in track T, and k is the number of trends that are extracted from the audio track. At the end of this process, T is summarised into C, and C is used in further classification steps. Fig. 2 depicts this procedure.

A richer description of the track is achieved as k increases. On the other hand, the total number of instances also increases, rising the demand for computing power for model training and feature storage. An optimal value for k depends on the dataset,

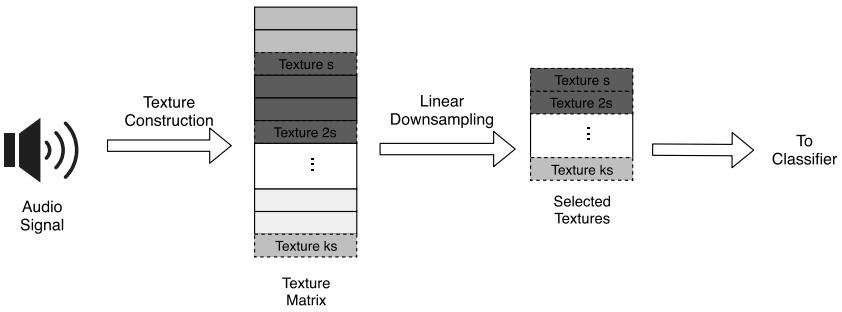


Fig. 3. LINSPACE Texture Selection. Texture construction is detailed in Fig. 1. Then, the track texture matrix is downsampled by selecting linearly-spaced textures.

as different datasets may have classes with varying degrees of overlapping. In this work, we evaluate different values for k to determine the best compromise between classification performance and computing power required for model training.

We call this method **K-Means Texture Selection** (KMEANSC). Because K-Means tends to estimate centroids that are distant from each other, we can expect that matrix C contains information that tends to be diverse. As centroids are more likely to be located around denser point clouds, they are also likely to represent typical textures. As a result, the C matrix highlights diverse typical sounds within the track. To the best of our knowledge, K-Means has not been used before to select representative textures for music genre classification.

3.2. Texture selection methods

To determine the effectiveness of our proposed K-Means texture selection method, we have compared it to other texture selection algorithms in the context of automatic music genre classification. Four competing methods were evaluated: LINSPACE, RANDOM, ALL, and FTS.

The five texture selection evaluated in this have distinct computational workload requirements. Hence, evaluating the resulting classification performance can guide the process of choosing the most appropriate texture selector, considering the task size and computing capabilities available.

3.2.1. Linear texture downsampling — LINSPACE

A number of previous research selects textures by linearly downsampling textures from each track [5]. Some variations of this method have been used before by several authors such as in [5–12]. A common strategy is to pick k linearly-spaced textures along the time axis. This procedure is shown in Fig. 3. Although previous research have used this strategy, it was not explicitly named. We refer to it in the rest of the paper as **LINSPACE Texture Selection**. Mathematically, a track texture matrix with m textures and n features $T \in \mathbb{R}^{m\times n}$, $T = [t_0, t_1, \ldots, t_m]^T$, $t_i \in \mathbb{R}^n$, is summarised by LINSPACE resulting in a matrix $L \in \mathbb{R}^{k\times n}$ such that $L = [t_s, t_{2s}, \ldots, t_{ks}]^T$, where $s = \left\lfloor \frac{m}{k} \right\rfloor$ and k is a parameter that controls the granularity of the downsampling procedure.

As k increases, more feature vectors are used for model training and testing. As a result it is expected that system performance improves as the k increases. However, as k increases, so does the required computing power for training and testing the machine learning models. Thus, k must be tuned for a suitable compromise.

LINSPACE does not use content to select representative textures for genre classification. In contrast, the proposed K-Means Texture Selection aims to select representative textures based on the premise that there are acoustic trends that are likely to appear in other tracks of the corresponding genres.

3.2.2. Random texture selection — RANDOM

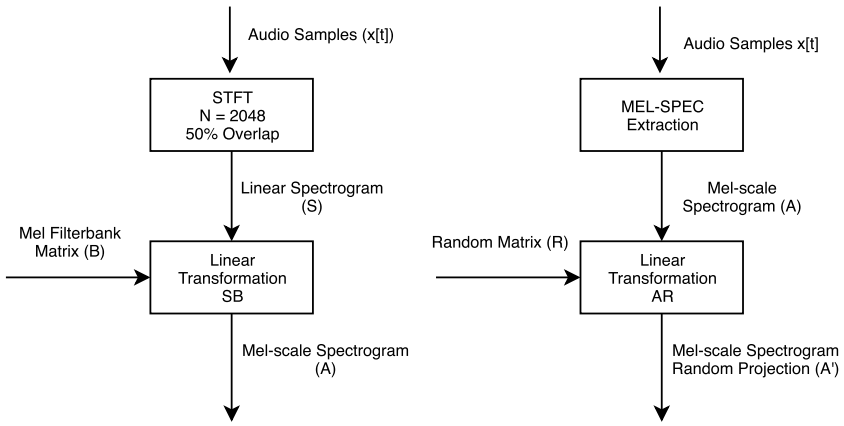
K-Means clustering begins by setting centroids to random points in the dataset. After the K-Means optimisation procedure, the clusters represent the trends in data. Considering the often high-dimensional texture space, it is interesting to verify how much the K-Means optimisation phase improves the selection of representative textures. Hence, we also evaluated a random texture selection procedure (RANDOM). Given a texture matrix $T \in \mathbb{R}^{m\times n}$, $T = [t_0, t_1, \ldots, t_m]^T$, $t_i \in \mathbb{R}^n$ where m is the number of textures and n is the number of features, RANDOM results in a matrix $T' \in \mathbb{R}^{k\times n}$ where $T' = [t_{p_0}, t_{p_1}, \ldots, t_{p_k}]$ and $P = [p_0, p_1, \ldots, p_{m-1}] = \text{RandomPerm}([0, 1, \ldots, m-1])$. RandomPerm generates a random permutation of the input vector based on a uniform distribution. k is a system parameter that should be tuned for a suitable compromise between additional computation and classification performance. Like LINSPACE, this description does not select representative textures based on content. However, the distribution of textures is different when compared to LINSPACE, as it selects textures uniformly over time, instead of following a linearly-biased distribution. We are not aware of any previous research that uses RANDOM texture selection for music genre classification .

3.2.3. All textures — ALL

To evaluate the impact using subsets of the texture matrices to describe tracks, we also investigated using all textures to represent each track. In this approach, no texture selection takes place. Given a texture matrix $T \in \mathbb{R}^{m\times n}$, where m is the number of textures and n is the number of features, ALL results in a matrix $T' = T$. Naturally, using all textures in the dataset for model training leads to higher computational costs. Comparing the results of the other texture selection strategies to ALL allows us to determine the relationship between the number of textures per track and classification performance. It also allows us to verify how representative the subset of textures is with respect to the full collection of textures. In other words, how well the subsets selected by the other methods are able to summarise the feature space spanned by all textures in the dataset.

3.2.4. Full-track statistics — FTS

As a baseline, we compare the results of the texture selection methods to full-track statistics (FTS). This method has been used in previous research [1,14,13]. It works by representing the entire track with a single texture. Given a texture matrix $T \in \mathbb{R}^{m\times n}$, $T = [t_0, t_1, \ldots, t_n]$, $t_i \in \mathbb{R}^m$, where m is the number of textures and n is the number of features, FTS is given by $T' \in \mathbb{R}^{1\times n}$ where $T' = [\mathbb{E}(t_0), \mathbb{E}(t_1), \ldots, \mathbb{E}(t_n)]$ and $\mathbb{E}(x)$ is the statistical average operator. By construction, this description considers that tracks can be described by an average texture. It offers a compact description and can be calculated efficiently. However, it disregards the fact that music is often not homogeneous, leading to a description that is unable to represent texture variation to its full-extent.



(a) MEL-SPEC Feature Set (b) MEL-RP Feature Set

Fig. 4. Low-level Feature Sets. Both MEL-SPEC and MEL-RP are low-level feature sets because they merely transform the input from the time-domain into the frequency domain. They are not meant to capture any perceptual audio or musical concepts.

3.3. Feature sets

To verify whether the trends in classification performance with texture selection depends on the feature set, we evaluated four distinct feature sets. These feature sets can be divided into three groups: "raw" features (MEL-SPEC, MEL-RP), handcrafted features, and features learned from data (MEL-AE). They are described below.

3.3.1. Mel-scale spectrograms

Two feature sets derive directly from audio data, and were used as "raw" features: MEL-SPEC and MEL-RP. The MEL-SPEC feature extraction is shown in Fig. 4(a). It starts with a track signal in the time domain. A STFT is calculated in frames of 2048 samples, with 50% overlap. The absolute value of the STFT is calculated, resulting in a magnitude matrix ($S \in \mathbb{R}^{t\times 1024}$). Then, a 128-bin Mel Filterbank matrix ($B \in \mathbb{R}^{1024\times 128}$) is applied to transform S into ($A = SB$). We call $A \in \mathbb{R}^{t\times 128}$ a Mel-Scale spectrogram (MEL-SPEC). The motivation for using this feature set relies on previous research, which has shown that Mel-Scale spectrograms highlight important aspects of the audio spectrum relevant to genre classification [18].

Many systems that rely on feature learning have been proposed to work with raw inputs such as MEL-SPEC [19,18,20]. We used it to evaluate how such texture descriptions can be used in a scenario where MEL-SPEC is used directly as a feature set, not as input to a feature learning mechanism. We assume that such inputs roughly model our timbre perception by highlighting the magnitude of spectral contents. Thus, it is expected that the Euclidean distance of such texture vectors relates to music content similarity.

3.3.2. Mel-scale spectrogram random projection

Another feature set derived directly from data is the random projection of Mel spectrograms (MEL-RP). MEL-RP feature extraction is shown in Fig. 4(b). The first step is to calculate the Mel-Scale Spectrogram A for the music track, as shown in Fig. 4(a). Then, a linear transformation $R \in \mathbb{R}^{128\times z}$, is applied to A, resulting in $A' = TR$. The number of columns in R, z, is lower than in A, which results in a projection into a lower-dimensional space. As R is a random matrix sampled from a Gaussian distribution with mean 0 and variance 1, we call $A' \in \mathbb{R}^{t\times z}$ the Mel-Scale Spectrogram Random Projection (MEL-RP). MEL-RP aims at transforming each texture in T into a stable embedding, preserving the underlying distance topology. This allows texture classification [21] in the projected space, instead of the Mel-Scale spectrogram space.

The effectiveness of random projections for dimensionality reduction is well-known in machine learning literature [22]. The Johnson–Lindenstrauss (JL) lemma [23] states that a random matrix $R \in \mathbb{R}^{N\times M}$, when $N > M$, projects the points in $A \in \mathbb{R}^{T\times N}$ into a stable embedding with high probability if $M = O(\log(T)\epsilon^{-2})$, $\epsilon \in (0, 1)$. N is the data original dimensionality, M is the target dimensionality, and T is the number of points in A. The ϵ constant quantifies the distortion introduced by the random transformation. It follows that, as more distortion is allowed, the smaller M can be. Thus, ϵ is a parameter that can be implicitly adjusted according to the application by optimising M in relation to a classification system performance measure.

Since the dimension of the embedding domain is lower ($M < N$), the computational effort needed for training a machine learning model in $\mathbb{R}^M$ is expected to be smaller than a model in $\mathbb{R}^N$. Furthermore, for suitable classifiers, the curse of dimensionality can be alleviated by transforming data points into a lower-dimensional space.

To the best of our knowledge, no previous research has employed MEL-RP as a feature set for genre classification. A similar idea, proposed by Choi [19], consists in setting the weights of a convolutional neural network to random values. The classification results are used as baseline for other approaches. The main difference is that the architecture is made up of convolution operations followed by non-linear activation functions. Choi has reported satisfactory results with this method. Furthermore, other authors have used random projections as a dimensionality reduction tool in further related works [24–26].

3.3.3. Handcrafted features

A variety of handcrafted features have been proposed for music classification [1,11,13,24]. We selected a set of features that are commonly used in the literature. We refer to the set we used as HANDCRAFTED. Handcrafted features are based on specialist knowledge. Therefore, they can be seen as features at a higher abstraction level than the "raw" feature sets presented above. In the context of our work, we are interested in how handcrafted features work in tandem with texture selection.

The handcrafted features selected are calculated for every frame from the STFT magnitude spectrum. First, the audio input signal sampled at 44 Khz is centred at mean 0, variance 1, and multiplied by a Hanning window in the time domain. Then, a

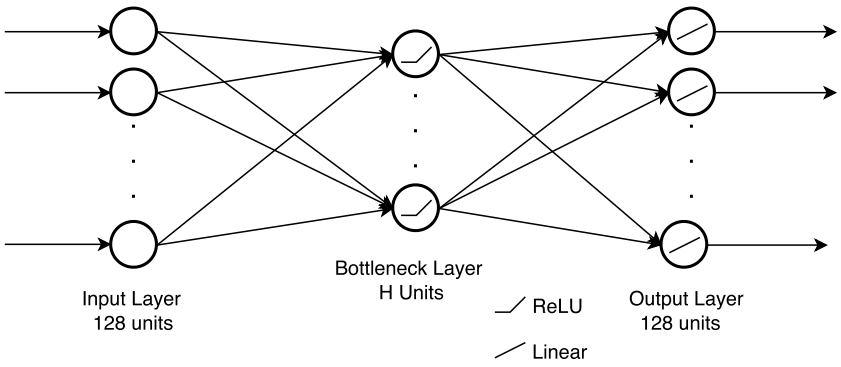


Fig. 5. Autoencoder Architecture. This network is designed to non-linearly encode the input signal into a H-dimensional latent space. The input signal can be linearly reconstructed. For reconstruction purposes, the latent space encoding retains much of the original information, allowing signal classification.

2048 sample STFT is calculated with 50% overlap over consecutive frames.

The following equations describes the features in our HANDCRAFTED feature set. $M[f]$ is the magnitude of the FFT of a given frame at frequency bin f, and N is the total number of frequency bins. [1] describes the **Spectral Centroid** as a measure of spectral brightness. It is calculated by:

$$C = \frac{\sum_{f=1}^{N} fM[f]}{\sum_{f=1}^{N} M[f]}$$

Spectral Rolloff is a measure of spectral shape [1]. It corresponds to the value R in the following equation:

$$\sum_{f=1}^{R} M[f] = 0.85 \sum_{f=1}^{N} M[f]$$

The **Spectral Flux**, which is a measure of spectral variation [1], can be calculated by:

$$F = \|M[f] - M[f-1]\|_2$$

Energy is a measure of signal strength, is calculated by:

$$E = \sum_{f=1}^{N} M[f]^2$$

Spectral Flatness is a measure of noise in an audio signal [27], and it is calculated by:

$$L = \frac{\exp\left(\frac{1}{N}\sum_{f=1}^{N} \ln\left(M[f]\right)\right)}{\frac{1}{N}\sum_{f=1}^{N} M[f]}$$

Zero Crossing Rate is another measure of noise in an audio signal. It is calculated over the time-domain signal, $x[i], | i = \{1, 2, \ldots, T\}$:

$$Z = \frac{1}{2T}\sum_{t=1}^{T} |\text{sign}(x[t+1]) - \text{sign}(x[t])|$$

where $\text{sign}(k) = 1$ if $k \geq 0$, otherwise 0.

MFCCs (Mel-Frequency Cepstral Coefficients) [28] are also part of the HANDCRAFTED feature set. MFCCs are widely used by the speech recognition and music information retrieval research communities. In MIR, they are commonly used as timbre descriptors.

In total, there are 26 features in the HANCRAFTED feature set: Spectral Centroid, Spectral Rolloff, Spectral Flux, Spectral Flatness, Energy, Zero Crossing Rate and the first 20 MFCC coefficients. We are interested in how much these rather simple features work along our proposed texture selection. Specifically, we want to verify how the classification results compare to more sophisticated feature sets, such as an autoencoder representation, discussed next.

3.3.4. Mel-scale autoencoder

An autoencoder is a neural network that learns a non-linear mapping from the input to the input itself [29]. A well-known architecture is based on a series of fully connected neuron layers. The middle layer is known as the *bottleneck*, is also known as the latent representation. The bottleneck usually has a lower dimensionality than the other layers. The bottleneck represents a compressed form of the input signal, hence transforming the input into a lower dimensionality vector.

In contrast to the other feature sets presented above, autoencoders provide features learned directly from data. Fig. 5 shows the architecture of the autoencoder used in this paper. Mel Spectrograms are the input to the autoencoder, hence its output as well. The number of neurons in the bottleneck is a parameter (H). A non-linear ReLU activation function is used in the bottleneck layer, forcing a non-linear projection of the input. The output layer uses a linear activation function, thus forcing a linear reconstruction of the input signal in the output space. This is an attempt to promote linear separability among different clusters of similar points. The bottleneck layer activations are used as features. We call this feature set MEL-Autoencoder (MEL-AE).

In preliminary tests we experimented with different autoencoder architectures. The final mean squared error was evaluated for different architectures and the results differed within small error margins. We chose the simplest architecture in terms of number of neurons and number of layers.

3.4. Feature selection

Feature selection is a common component in many machine learning systems. Previous research showed that it has a positive effect on music genre classification [30]. In this paper, we investigate how feature selection works along with texture selection. We are not aware of any previous research that has explored this relationship.

We chose to use a simple univariate correlation filter for feature selection known as ANOVA feature selection. First, a Pearson correlation coefficient is calculated for each feature with respect to the training labels. Then, only the features corresponding to the highest scoring coefficients are used for model training and testing. The number of features to keep is a parameter. We chose this feature selection method because it is well-known and has an intuitive behaviour which makes analysis straightforward. As a baseline, we also evaluated texture selection without any feature selection.

4. System architecture

In this section we present experiments that were designed to investigate how texture selection impacts music genre classification performance. Specifically, we are interested in how our

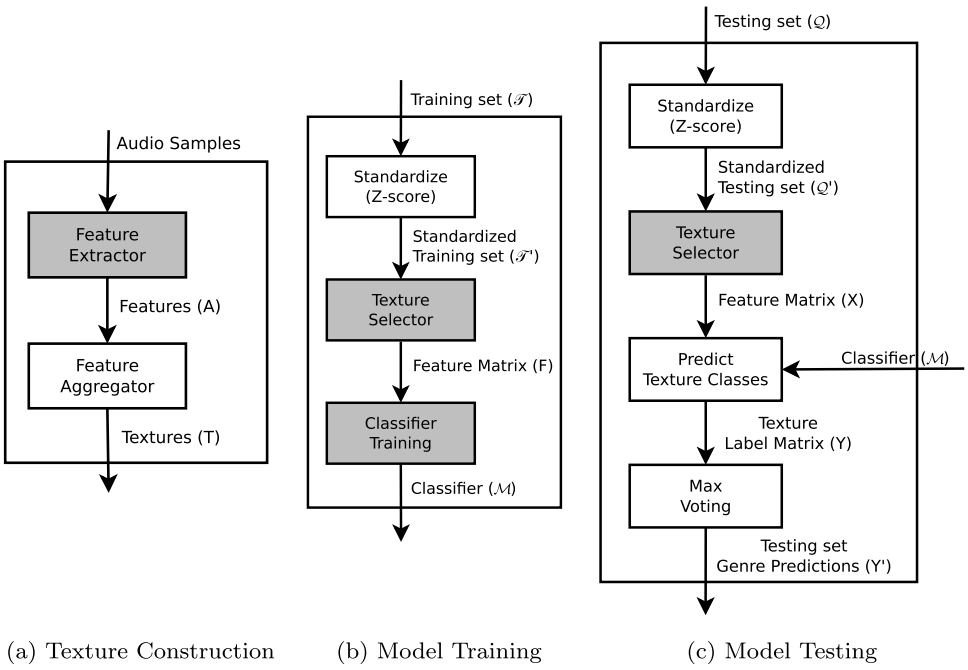


Fig. 6. Proposed System Architecture. The proposed architecture consists of (a) Texture Calculation, (b) Model Training, and (c) Model Testing. Textures are constructed as in (a) for all tracks in the Training and Testing sets. (b) shows the training pipeline and its output, the trained classifier ($\mathcal{M}$). (c) shows the testing pipeline, which outputs predictions for the tracks in the testing set using the classifier trained in (b).

method based on the K-Means clustering algorithm compares to other methods, namely LINSPACE, RANDOM, ALL, and FTS. As stated in Section 3, we also investigate how other components of the classification pipeline affect system performance, such as feature selection and the feature set used to describe textures.

The system architecture is shown in Fig. 6. It consists of three stages: Texture Construction (Figs. 1 and 6(a)), Model Training (Fig. 6(b)), and Model Testing (Fig. 6(c)). The shaded boxes indicate the system components that were evaluated with different parameters and techniques. These variations are presented in Section 4.1.

Texture Construction (Figs. 1 and 6(a)) begins by extracting features over 23 ms frames from music tracks sampled at 44KHz. For each track, a feature matrix (A) is calculated, consisting of a j-dimensional low-level feature vector for every frame. The first and second-order deltas are calculated and concatenated into the low-level feature vector, resulting into a $3j$-dimensional feature vector per frame. Textures are then constructed by aggregating 216 consecutive low-level feature vectors, resulting in m textures that cover approximately 5s of the track. Low-level feature vectors are aggregated into textures by average and standard deviation, resulting in n-dimensional textures, where $n = 6j$. Successive textures are calculated every 10 frames, resulting in a 10× downsample and around 95% overlap between textures. The resulting texture matrix ($T \in \mathbb{R}^{m \times n}$) is calculated for every track in the dataset.

The results of the texture construction stage are input into to a machine-learning system. It is divided into two stages: model training and model testing. The model training stage uses a part of the dataset to estimate its latent parameters. The model testing stage uses another, held-out part of the dataset to evaluate the classification performance.

The input to the Model Training stage (Fig. 6(b)) is the training set ($\mathcal{T}$), which is the set of texture matrices corresponding to the training tracks. First, textures are standardised feature-wise to mean 0, standard deviation 1, resulting in the standardised training set $\mathcal{T}'$. The standardisation parameters are saved to be applied later to the testing set. Texture selection is performed on every texture matrix of the training set independently. The number of desired textures per track is a parameter, which we call k. The texture selector outputs a training Feature Matrix (F), which concatenates all selected textures from every track in the training set. Assuming all selected textures are representative of the genre, every texture from the same track are assigned the track label. A classifier is trained with the samples in F, along with the corresponding labels. The classifier ($\mathcal{M}$) can be seen as a function that maps each texture to a label.

The Model Testing stage (Fig. 6(c)) receives a testing set ($\mathcal{Q}$) as input, which is the set of texture matrices corresponding to the testing tracks. The same standardisation parameters applied to the training set $\mathcal{T}$ are applied to $\mathcal{Q}$, resulting in a standardised testing set ($\mathcal{Q}'$). Similarly to the training stage, texture selection is performed on each track independently for each texture matrix in the testing set. The same number of textures per track are used in both training and testing stages. The texture selector outputs a testing matrix (X), with every selected texture of each track in the testing set. Then, $\mathcal{M}$ is applied row-wise to X, resulting in a Texture Label Matrix (Y), with a label prediction for every selected texture of every track in the testing set. A final classification is decided for each track by majority voting. The output for the entire testing set is the prediction matrix (Y').

4.1. System parameters

Each shaded box in Fig. 6 indicates system components that we evaluated with different parameters and techniques. The techniques correspond to the different texture selection methods, feature sets, and feature selection, detailed in Sections 3.1, 3.2, 3.3, and 3.4. Each combination of techniques results in a different classification system. Most techniques have free parameters that have to be set, and result in different outcomes. In this Section we present the parameters evaluated for all techniques we investigated. With this, we aim to make our work reproducible.

Table 1
Texture Selection Parameters. We evaluated the classification performance impact of KMEANSC, LINSPACE and RANDOM texture selectors with an increasing number of selected textures.

Texture selectors	Parameters	Values
KMEANSC LINSPACE RANDOM	# of Textures (K)	{5, 20, 40}
ALL	–	–
FTS	Aggregation	MEAN + STDEV

Table 2
Learning Algorithm Parameters. The parameters evaluated during the hyperparameter tuning phase are shown for both algorithms used in the experiments.

Learning algorithm	Parameter	Values
SVM	C	{1, 10, 100, 1000}
	γ	$\frac{1}{n_{features}}$
	Kernel	rbf
KNN	K	{1, 3, 5, 7, 9}

4.1.1. Texture selection parameters

Table 1 shows the parameters evaluated for all texture selectors. A single parameter is evaluated for KMEANSC, LINSPACE, and RANDOM: the number of textures used to describe each track. The exact values were chosen empirically, based on preliminary experiments. A low number such as 5 textures allows us to evaluate which selector is able to achieve acceptable classification performance with lower computing power requirements. A good classification with few textures also gives us an idea of the relevance of the selected textures with respect to the target genres. We also evaluate the texture selectors with an increasing number of textures. This allows us to determine the relationship between the number of textures per track and classification performance.

The ALL selector increases the computing costs, specially for SVM model training. Table 4 shows the total number of textures in each dataset. To make large-scale SVM training tractable, we used the ThunderSVM implementation [31]. It uses a GPU-based solver that speeds up SVM training by up to 3 orders of magnitude, depending on the size of the dataset.

4.1.2. Learning algorithms and feature selection parameters

We evaluated the systems with two popular learning algorithms, namely SVM and KNN. The Support Vector Machine (SVM) is a well-known learning algorithm used in various machine learning music applications. It relies on mapping points into a higher dimensionality space where they become linearly separable by a hyperplane [32]. Thus, by design, SVM is not only robust to high-dimensional data, but projecting data into high-dimensional spaces is part of its strategy to solve the pattern classification problem. Two regularisation parameters responsible for softening the decision boundary are usually optimised, namely C and γ. Table 2 shows the SVM parameters evaluated in the experiments. The main drawback of the SVM is the time complexity of its training algorithm, which depends on the number of training samples, the dimensionality of the training vectors and the type of kernel.

The K-Nearest Neighbours (KNN) is another well-known learning algorithm. It uses a distance metric, commonly Euclidean distance, to measure similarity between points, in which similarity is inversely proportional to distance. To label an unknown point, KNN first calculates its k nearest points. Then, a majority vote with the corresponding labels is used make the final decision. Table 2 shows the values for K we evaluate in our experiments. A drawback of KNN is that it suffers from the curse of dimensionality, resulting in degraded classification performance in high-dimensional spaces.

Table 3
Feature Extraction Parameters. The parameters and respective values for all feature sets evaluated are shown. Values in brackets represent the set of values that were evaluated in the experiments.

Feature sets	Parameters	Values
MEL-SPEC	Mel Bins	128
	FFT Size	2048
	Hop Size	1024
	Window	Hanning
MEL-RP	Target Dim (M)	{8, 26, 51, 75, 100}
MEL-AE	Bottleneck Dim (H)	{16, 32, 64, 128, 256}
HANDCRAFTED	FFT Size	2048
	Hop Size	1024
	MFCC Coeff.	20
	Window	Hanning

The number of selected features is a parameter for the ANOVA feature selection algorithm. As the number of features in each feature set varies, as shown next, we decided to evaluate fractions of the total number of features. The following fractions of the highest scoring features were evaluated: 20%, 40%, 60%, and 80%. We also evaluated all systems with no feature selection.

4.1.3. Feature set parameters

The Feature Extractor parameters evaluated are shown in Table 3. The MEL-RP and MEL-AE input spectrograms are calculated with the same parameters as MEL-SPEC. HANDCRAFTED features are calculated from the STFT instead.

The target dimensionality for MEL-RP is a parameter. Four target dimension sizes for MEL-RP were chosen linearly between 26 and 100. To get an idea of how the system performed at a very low dimensionality, we also evaluated the system with only 8 random projection features. For MEL-AE, five different bottleneck sizes were evaluated.

Next, we describe the datasets used in the experiments.

5. Datasets

The texture selection strategies were evaluated with four different publicly available datasets: GTZAN [1], ISMIR [33], LMD [34], and HOMBURG [35]. We chose these datasets because each one presents particular challenges. Evaluating system performance on datasets with different challenges leads to a better understanding of how effective the texture selection methods are under different situations.

The datasets are well-known in the music information retrieval community. Table 4 summarises the datasets and shows their diversity regarding genres, number of tracks, track length, total number of textures, and average number of textures per track.

GTZAN [1] was one of the first widely available datasets for MGC and it is widely known in the research community. This dataset consists of 1000 music clips of 10 western music genres. GTZAN is balanced among its 10 genres (blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, rock), with 100 clips each. Each music clip is 30 s long. Its flaws are well-known by the research community. The main faults are: repeated tracks, multiple tracks from the same artist, cover tracks, mislabellings and extreme distortions [36]. To address these faults, we created a 3-fold dataset split using the findings in [36] as guidance. We call this split GTZAN-ARTF in the results section. Although Sturm shows that there are still some unidentified excerpts in GTZAN,

Table 4
Datasets used in the experiments. Four datasets were evaluated in the experiments. The datasets are varied with respect to the number of tracks, length of the audio excepts and the number of output classes (Music Genres). Six entries are shown, as two datasets were evaluated both as full-length tracks and 10 s excerpts.

Datasets	Genres	Tracks	Textures	Track length	Avg. textures per track
GTZAN [1]	10	1000	107K	30s	107
ISMIR [33]	6	1458	1,5M	FULL	1046
ISMIR (10 s)	6	1458	31K	10s	20
LMD [34]	10	1300	1,2M	FULL	940
LMD (10 s)	10	1300	27K	10s	21
HOMBURG [35]	9	1886	40K	10s	21

the proposed changes fixes most of the identified faults. The ARTF split we used is available online.[1]

This 3-fold split is artist-filtered, which means that tracks from same artist and genre do not appear in both training and testing sets. It has been shown that artist-filtered splits makes the genre classification problem significantly harder [16,37]. As a consequence, the classifier performances are expected to be lower. One of the reasons for this is that the classifier can learn patterns that relate to specific artists, such as the voice timbre of the lead singer, in contrast to patterns relating to genre, regardless of artist-specific characteristics. Not rarely, an artist works solely on a single genre, thus all of their tracks are labelled with the same genre. Hence, a classification system may correctly predict a genre to a track based on artist-specific patterns, instead of genre-defining patterns. This causes songs from the same artist in the test set to be easier to classify, but this does not mean that the classifier has generalised well to other artists performing on the same genre.

GTZAN is a widely used dataset for evaluating genre classification systems [38]. Often, a random 10-fold split is used, hence ignoring the artist filtering problem. For comparison purposes, we also included the results with this random split. We call this split GTZAN-RANDOM.

LMD (Latin Music Dataset) [34] is another well known dataset that consists of popular Latin-American music genres. This dataset was originally labelled according to dancing patterns, specially for deciding between genres that are similar regarding timbre, such as axé/pagode and gaúcha/sertaneja. We used an artist-filtered subset of LMD in our experiments, which avoids the same problems present in the GTZAN dataset and allows a balanced 3-fold split. We used the same split as in [39,5]. This subset consists of 1300 full-length music tracks across 10 different genres, each with 130 tracks. The split we used is available online.[2]

The ISMIR 2004 [33] dataset consists of 1458 full-length music tracks across 6 western genres. We used the train/test split supplied in the ISMIR 2004 challenge homepage, with 729 tracks in both the training and testing sets. No artist filter was applied to this dataset because the artist information is not available for all tracks, only for tracks in the training set. This dataset is not balanced, thus the number of tracks is not the same for all classes. The majority of the tracks in the training set are *classical* (320), followed by *world* (122), *electronic* (115), *rock_pop* (101), *metal_punk* (45), and *jazz_blues* (26). The number of tracks per genre in the testing set is similar to the training set.

HOMBURG [35] is a lesser known dataset that also consists of western music genres. It comprises 1886 music clips across 9 genres. This dataset is interesting for evaluating texture selection techniques, since the tracks are only 10 s long. We used a standard 10-fold random split for cross validation. We did not apply an artist filter to HOMBURG because the number of tracks per artist is low, with an average of 1.28 tracks per artist. It is also quite unbalanced, ranging from 504 tracks for Rock to 47 for Funk/Soul/RnB. The remaining genres have 190 ± 90 tracks each.

[1] https://github.com/julianofoleiss/gtzan_sturm_filter_3folds_stratified/
[2] https://github.com/julianofoleiss/lmd_3f_stratified_artist_filter/

6. Results

Average F1-scores and standard deviations for all experiments are shown in the Appendix, in Figs. A.9, A.10, A.11, A.12, and A.13. However, to make it easier to understand the results, we report percent changes over the baselines for all texture selectors evaluated.

Considering the stochastic behaviour in the KMEANSC texture selector resulting from the randomness of the initial centroid candidates, all the experiments were performed three times. We report the average F1-scores over the three trials, along with the respective standard deviations.

To increase readability, we use the word "significant" exclusively to mean "statistically significant". Unless otherwise stated, we used a two-tailed paired Student's T-test to test for statistical significance between any two sample measurements. We considered a p-value threshold of 5% for rejecting the null-hypothesis. Also regarding statistical wording, we use the word "comparable" to mean "a difference lower than two standard deviations".

6.1. Learning algorithms

6.1.1. FTS baseline

Tables 5a and 5b shows the percent change of the F1-Score over the FTS baseline for KMEANSC, LINSPACE, and RANDOM texture selectors. These tables summarises the percent changes over FTS shown in Figs. A.9a, A.9c, A.10a, A.10c, A.11a, A.11c, A.12a, A.12c, A.12e, A.12g, A.13a, A.13c, A.13e, and A.13g.

The results are shown for the MEL-AE feature set. Both tables show that as the number of textures per track increases, the improvements over the baseline also increases. This is true for both classifiers and for all texture selectors. For the datasets with shorter tracks, this increase is more modest.

With 5 textures per track, the KMEANSC texture selector is the only one that is able to improve over the baseline on both full-length datasets, LMD and ISMIR. This is true for both SVM (Table 5a) and KNN (Table 5b). Table 5a shows that with 20 textures per track and the SVM classifier, KMEANSC, LINSPACE and RANDOM improve over the FTS baseline on both full-length datasets. In this case, the improvements with KMEANSC are significantly higher than LINSPACE and RANDOM. Table 5b shows that with KNN, there are also improvements with 20 textures per track for all texture selectors. The improvements with KMEANSC are also higher than the improvements with LINSPACE and RANDOM. However, both LINSPACE and RANDOM show no improvement for the LMD dataset with KNN.

Table 5a shows that for the 30 s datasets, GTZAN-RANDOM and GTZAN-ARTF, the improvements with all texture selectors over the FTS baseline with SVM are similar and not statistically different. This is true with both 5 and 20 textures per track. The results with 20 textures are significantly better than 5 textures, although the difference is smaller than observed with the full-length datasets. The improvements occurred in both the non-artist-filtered and artist-filtered versions of the GTZAN dataset.

Table 5
Percent change in F1-Score from the FTS baseline with the MEL-AE feature set. The baseline column shows the average F1-score and standard deviation. The KMEANSC, LINSPACE, and RANDOM columns show the percent change over the average of the baseline F1-Score for the three texture selectors. 5 and 20 refer to the number of textures per track that were used for both model training and testing. (a) Results with SVM. (b) Results with KNN.

(a) SVM

Dataset	FTS Baseline (F1-Score)	KMEANSC ($\%\Delta_{FTS}$)		LINSPACE ($\%\Delta_{FTS}$)		RANDOM ($\%\Delta_{FTS}$)	
		5	20	5	20	5	20
LMD	0.77 ± 0.03	3.9	9.1	−6.5	6.5	−6.5	3.9
ISMIR	0.85 ± 0.01	2.4	5.9	−2.4	3.5	−3.5	3.5
GTZAN-RANDOM	0.74 ± 0.04	5.4	8.1	4.1	8.1	4.1	8.1
GTZAN-ARTF	0.49 ± 0.03	16.3	20.4	12.2	16.3	10.2	18.4
HOMBURG	0.52 ± 0.03	5.8	7.7	5.8	7.7	5.8	7.7
LMD-10s	0.59 ± 0.04	13.6	13.6	11.9	13.6	11.9	15.3
ISMIR-10s	0.76 ± 0.02	−1.3	1.3	0.0	1.3	0.0	2.6

(b) KNN

Dataset	FTS Baseline (F1-Score)	KMEANSC ($\%\Delta_{FTS}$)		LINSPACE ($\%\Delta_{FTS}$)		RANDOM ($\%\Delta_{FTS}$)	
		5	20	5	20	5	20
LMD	0.64 ± 0.01	7.8	12.5	−20.3	−9.4	−26.6	−10.9
ISMIR	0.77 ± 0.03	2.6	7.8	−10.4	2.6	−10.4	0.0
GTZAN-RANDOM	0.63 ± 0.05	7.9	11.1	−1.6	6.3	−4.8	4.8
GTZAN-ARTF	0.41 ± 0.02	12.2	19.5	0.0	7.3	4.9	9.8
HOMBURG	0.42 ± 0.02	0.0	4.8	−2.4	2.4	−2.4	2.4
LMD-10s	0.47 ± 0.01	−2.1	2.1	0.0	0.0	−6.4	−2.1
ISMIR-10s	0.68 ± 0.03	2.9	2.9	2.9	4.4	−2.9	4.4

Table 6
Percent change in F1-Score from the ALL baseline with the MEL-AE feature set. The baseline column shows the average F1-score and standard deviation. The KMEANSC, LINSPACE, and RANDOM columns show the percent change over the average of the baseline F1-Score for the three texture selectors. 5 and 20 refer to the number of textures per track that were used for both model training and testing. (a) Results with SVM. (b) Results with KNN.

(a) SVM

Dataset	ALL Baseline (F1-Score)	KMEANSC ($\%\Delta_{ALL}$)		LINSPACE ($\%\Delta_{ALL}$)		RANDOM ($\%\Delta_{ALL}$)	
		5	20	5	20	5	20
LMD	0.83 ± 0.02	−3.6	1.2	−13.3	−1.2	−13.3	−3.6
ISMIR	0.89 ± 0.03	−2.2	1.1	−6.7	−1.1	−7.9	−1.1
GTZAN-RANDOM	0.81 ± 0.03	−3.7	−1.2	−4.9	−1.2	−4.9	−1.2
GTZAN-ARTF	0.59 ± 0.05	−3.4	0.0	−6.8	−3.4	−8.5	−1.7
HOMBURG	0.57 ± 0.03	−3.5	−1.8	−3.5	−1.8	−3.5	−1.8
LMD-10s	0.66 ± 0.01	1.5	1.5	0.0	1.5	0.0	3.0
ISMIR-10s	0.78 ± 0.03	−3.8	−1.3	−2.6	−1.3	−2.6	0.0

(b) KNN

Dataset	ALL Baseline (F1-Score)	KMEANSC ($\%\Delta_{ALL}$)		LINSPACE ($\%\Delta_{ALL}$)		RANDOM ($\%\Delta_{ALL}$)	
		5	20	5	20	5	20
LMD	0.58 ± 0.06	19.0	24.1	−12.1	0.0	−19.0	−1.7
ISMIR	0.83 ± 0.04	−4.8	0.0	−16.9	−4.8	−16.9	−7.2
GTZAN-RANDOM	0.68 ± 0.04	0.0	2.9	−8.8	−1.5	−11.8	−2.9
GTZAN-ARTF	0.44 ± 0.03	4.5	11.4	−6.8	0.0	−2.3	2.3
HOMBURG	0.42 ± 0.03	0.0	4.8	−2.4	2.4	−2.4	2.4
LMD-10s	0.47 ± 0.02	−2.1	2.1	0.0	0.0	−6.4	−2.1
ISMIR-10s	0.68 ± 0.03	2.9	2.9	2.9	4.4	−2.9	4.4

For KNN (Table 5b) the improvements are similar, although for 5 textures there was no improvement for both LINSPACE and RANDOM in the GTZAN-RANDOM dataset.

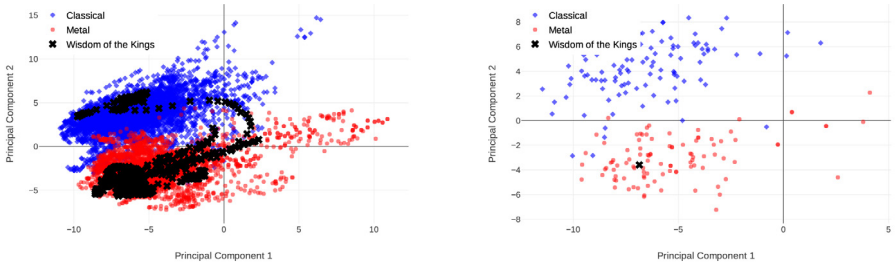
For the short, 10 s datasets (HOMBURG, LMD-10s, ISMIR-10s), Table 5a shows that the improvements for all texture selectors are similar. The difference in performance between 5 and 20 textures is smaller than they were with full-length datasets and 30 s datasets. This smaller difference is true for all three texture selectors. With KNN (Table 5b), the improvements from texture selection are more modest and there is no clear pattern of improvement.

6.1.2. ALL baseline

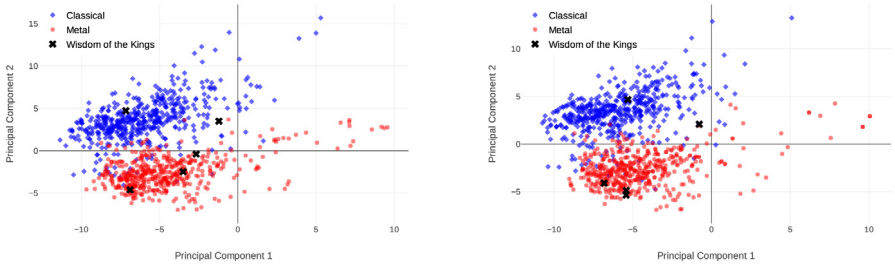
Tables 6a and 6b shows the percent change of the F1-Score over the ALL baseline for KMEANSC, LINSPACE, and RANDOM texture selectors. These tables summarises the percent changes over ALL shown in Figs. A.9a, A.9c, A.10a, A.10c, A.11a, A.11c, A.12a, A.12c, A.12e, A.12g, A.13a, A.13c, A.13e, and A.13g.

Table 6a shows that with 5 textures per track in the full-length datasets, the difference between ALL and KMEANSC is significantly smaller than the difference between ALL and LINSPACE and ALL and RANDOM. With 20 textures, this difference among is negligible and the three texture selectors achieve results statistically the same as the ALL baseline. In some cases the results with KMEANSC, LINSPACE and RANDOM are superior to the results with ALL, as shown by the percent changes greater than zero. Table 6b shows that with KNN the trends are similar to those with SVM. Moreover, the gains in the LMD dataset with KMEANSC for both 5 and 20 textures are larger than usual, achieving a 19% and 24.1% improvement over the ALL baseline. Such large improvement is not present with LINSPACE and RANDOM.

For the 10 s and 30 s datasets, the difference between ALL and each of the texture selectors is not significant with 5 textures per track. With 20 textures, the difference between ALL and the three texture selectors is even smaller. Table 6b shows that with KNN, KMEANSC improves over the ALL baseline in most scenarios,



(a) ALL (b) FTS



(c) KMEANSC (5 textures / track) (d) LINSPACE (5 textures / track)

Fig. 7. PCA projections of the selected textures of "Wisdom of the Kings", by Rhapsody. The figure shows the selected textures over the GTZAN dataset (HANDCRAFTED features). ALL shows clearly the excerpt has characteristics of both Classical and Metal genres. FTS correctly positions the track in the Metal region, although not representing its genre fusion. KMEANSC5 selects textures evenly among the textures in the track. LINSPACE tends to focus more on the most dense region.

either with 5 or 20 textures. With LINSPACE and RANDOM, the difference to ALL is not significant either, although the changes, even with 20 textures, are more modest than KMEANSC.

6.2. Feature sets

The results presented in Section 6.1 showed the results for the MEL-AE feature set. In general, the improvement patterns shown in Section 6.1 are also present in the other feature sets evaluated. As Figs. A.9, A.10, A.11, A.12, and A.13 show, most feature sets achieve comparable results in most cases. In general, the best average results were achieved with HANDCRAFTED and MEL-AE features. These results are followed by MEL-RP, and then by MEL-SPEC. Tables presenting the percent changes from both baselines with the HANDCRAFTED, MEL-SPEC and MEL-RP are available as supplementary materials.

We present the results (Fig. A.9, A.10, A.11, A.12, and A.13) for MEL-RP with 75 features and MEL-AE with 128 features. For both feature sets, the results improve in all datasets as the number of features increases with SVM and KNN and all texture selectors evaluated. The results with MEL-RP saturate at 75 features, while the results with MEL-AE saturate at 128 features.

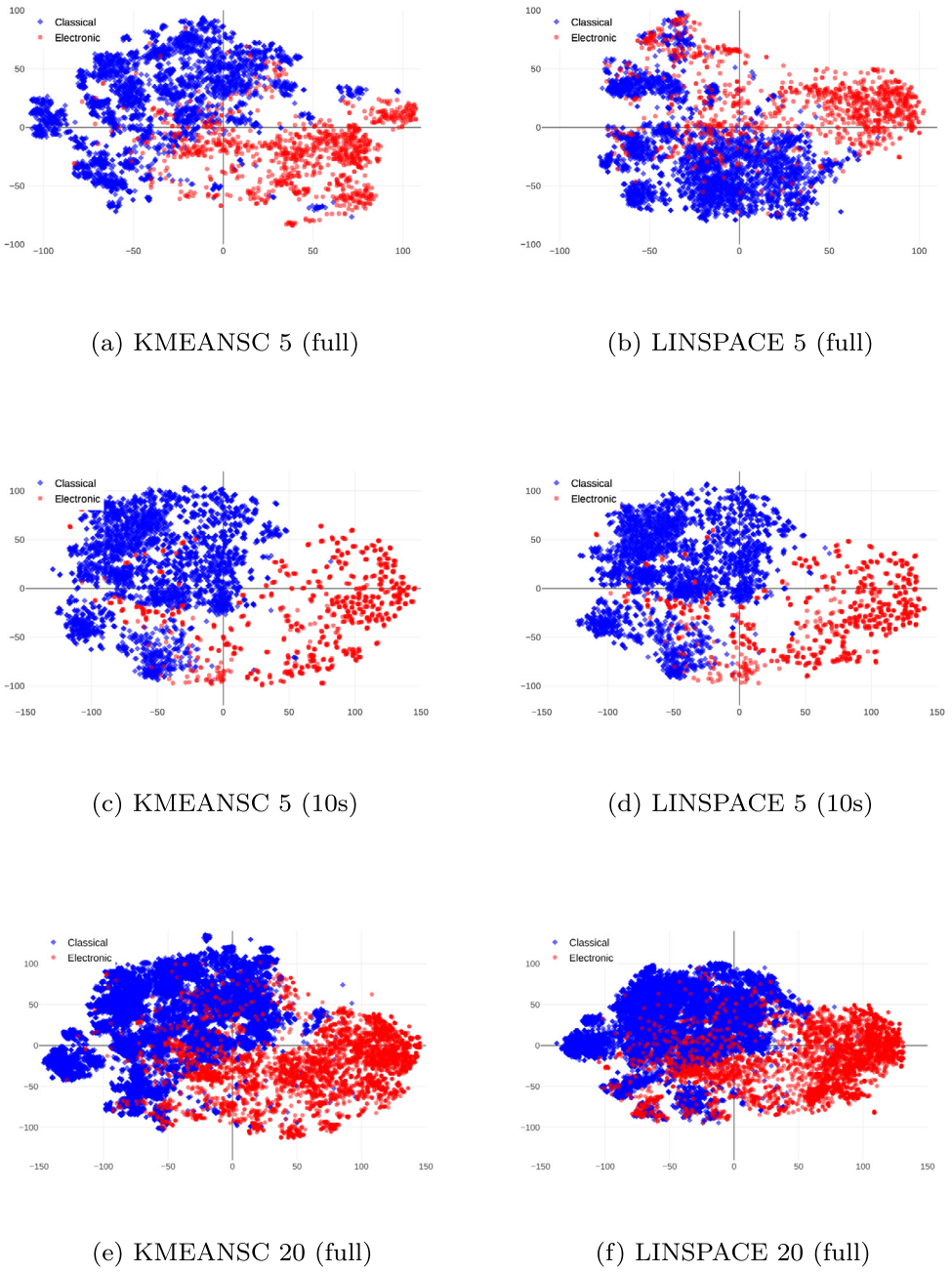
6.3. Feature selection

The right-hand side of Figs. A.9, A.10, A.11, A.12, A.13 shows the results of feature selection in all experiments. These results are directly comparable to the left-hand side of the Figures showing the results with no feature selection. We provide the actual changes as supplementary material. The improvements were similar for all feature sets, so we present an overall evaluation. Comparing the improvements over the no feature selection baseline by texture selector, there were no significant changes with SVM ANOVA in most cases for all selectors. In total, 22 out of 224 results changed significantly. For FTS, all 5 significant changes were negative, as well as the 2 changes in RANDOM. For ALL, the 4 significant changes were improvements. The 8 improvements in LINSPACE were also positive. For KMEANSC, 1 change was positive, while 2 were negative. The average performance improvement in significantly better cases was 4.90%±1.98, while the performance decrease was 4.59%±1.35 in significantly worse cases.

On the other hand, the results with ALL, LINSPACE and RANDOM changed significantly in many cases with KNN ANOVA. In total, 80 out of the 224 results changed significantly. For ALL, all significant changes were positive, improving the results in 12 out of 28 cases, mostly in the HANDCRAFTED and MEL-SPEC feature sets, while none of the cases with MEL-SPEC-AE changed. For LINSPACE, the changes were significant in 33 out of 56 cases, of which 29 were improvements. For RANDOM, the changes were significant in 25 out of 56 cases, of which 23 were improvements. The performance drop with RANDOM happened only with MEL-SPEC features. For KMEANSC and FTS, the changes were significant in 4 out of 56 cases each, of which 2 were improvements. The average performance improvement in significantly better cases was 14.08% ± 7.85, while the performance decrease was 7.06% ± 2.9 in significantly worse cases.

Comparing the improvements of the no-selection baseline by dataset, there were a few result changes per dataset with SVM ANOVA. In both GTZAN-RANDOM and LMD, there were 5 out of 32 significant changes, of which 4 were improvements. In ISMIR and ISMIR-10s, both significant changes were improvements. In GTZAN-ARTF, none of the changes were significant over the no feature selection baseline. For HOMBURG, there were 4 out of 32 significant changes, of which 1 was an improvement over the baseline. For LMD-10s, there were 4 out of 32 significant changes, which were all worse than the baseline.

In contrast to SVM ANOVA, most datasets had a larger number of significant changes with KNN ANOVA. In LMD, there were significant changes in 20 out of 32 cases, of which 19 were



(a) KMEANSC 5 (full) (b) LINSPACE 5 (full)

(c) KMEANSC 5 (10s) (d) LINSPACE 5 (10s)

(e) KMEANSC 20 (full) (f) LINSPACE 20 (full)

Fig. 8. t-SNE embeddings depicting feature spaces derived from texture selection over the ISMIR dataset (HANDCRAFTED features). The embeddings allow inspecting how KMEANSC and LINSPACE span the feature space. Although a significant difference is shown between (a) and (b), (c) and (d) are very similar. This shows that diversity within tracks is a key factor for determining the resulting texture selection feature space span.

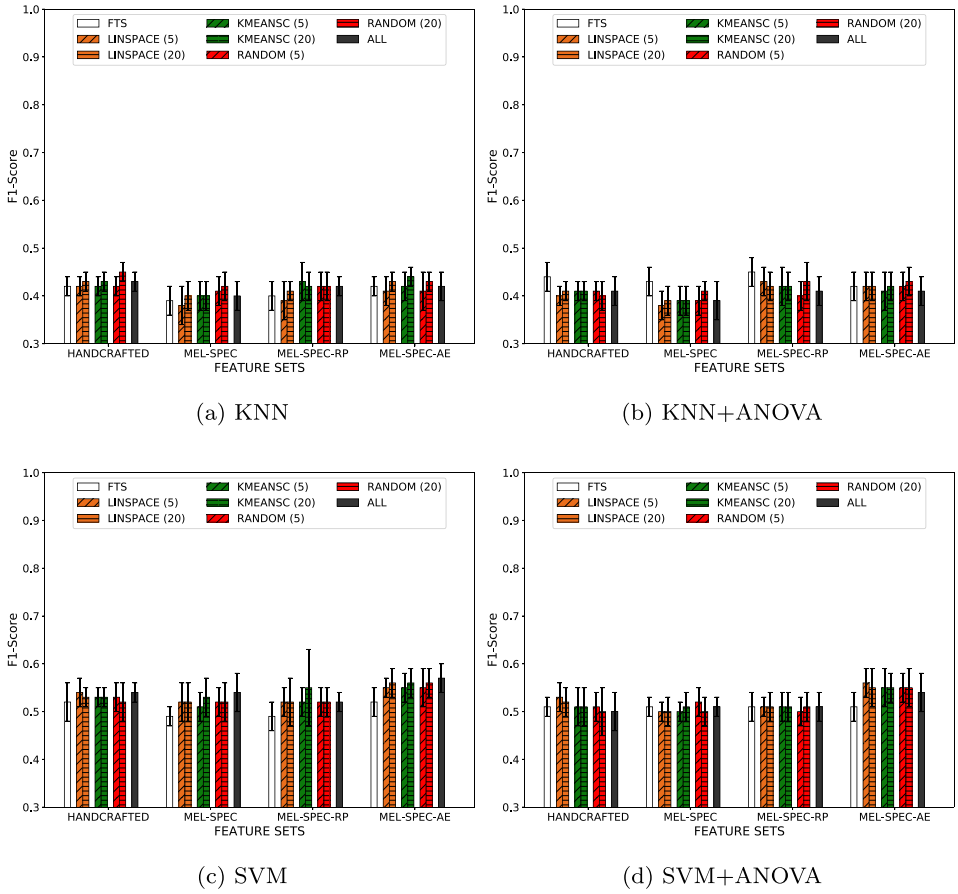
improvements. In ISMIR, there were changes in 11 out of 32 cases, all improvements. In GTZAN-RANDOM, there were significant changes in 15 out of 32 cases, all improvements. In GTZAN-ARTF, the changes were significant in 8 out of 32 cases, of which 7 were improvements. In HOMBURG, there were significant changes in 10 out of 32 cases, of which half were improvements. For LMD-10s, 13 out of 32 cases presented significant changes, of which 11 were improvements. Lastly, in ISMIR-10s the changes were significant in 3 out of 32 cases, of which 2 were improvements.

6.4. Feature space visualisation

We used two visualisation techniques to identify the effect of the different texture selection strategies on the resulting feature spaces. First, we use Principal Component Analysis (PCA) to identify the effects of the texture selection processes in the representation of a single track, as shown in Section 6.4.1. Then, we use t-SNE unsupervised manifold learning to identify the effects of texture selection on the resulting feature space, as presented in Section 6.4.2.

6.4.1. Texture selection visualisation of a single track

In this section we show how the selected textures of a single track are distributed in the feature space. For such, we used PCA projections of a single track textures over a subset of the GTZAN dataset feature space . We used the HANDCRAFTED feature set to describe textures. Fig. 7 shows PCA projections of the textures selected from the track “Wisdom of the Kings” by the Italian Heavy Metal band Rhapsody. This track was chosen because it



(a) KNN (b) KNN+ANOVA

(c) SVM (d) SVM+ANOVA

Fig. A.9. Average F1-score across 10 random Folds for the HOMBURG Dataset. Results for all texture selectors are shown. The colour bars show the results for 5, and 20 selected textures in applicable selectors.

has both typical Heavy Metal parts, with fast drums and guitars, and interludes composed in Classical style.

Fig. 7(a) shows that the textures of the track are distributed over the Classical and Metal regions of the feature space. This shows that although this track is most likely labelled as "Metal", it is a fusion between Metal and Classical. The FTS representation shown in Fig. 7(b) correctly positions the track close to the Metal genre. However, it discards the information related to the Classical segments.

KMEANSC selected textures that are more evenly distributed along the feature space, as shown in Fig. 7(c). KMEANSC selected textures from both the Classical and Metal regions. LINSPACE, on the other hand, selected textures that are more common throughout the track, as evidenced by the selected textures in the denser areas of Fig. 7(d).

To get an idea of the sounds corresponding to the textures selected by either KMEANSC and LINSPACE, we created audio clips that correspond to the 5s of audio closest to each texture. Audio Sample 1, presented as supplementary material, corresponds to the KMEANSC textures. This sample consists of audio segments that sound different to one another. Audio corresponding to LINSPACE textures are in Audio Sample 2, also presented as supplementary material. This sample consists of audio segments that represent the most common sounds in the track, which are more likely to be selected by linear downsampling.

6.4.2. Feature space visualisation of a dataset

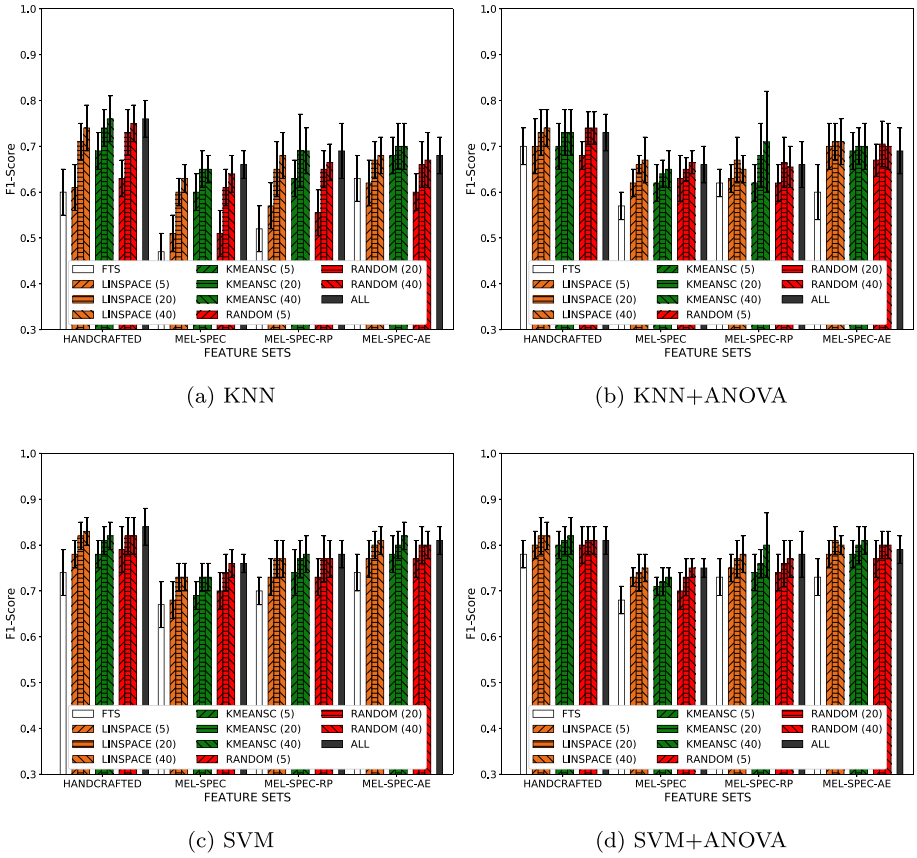
In this section we present feature spaces that were derived from the texture selection methods of the ISMIR dataset. For such, we used t-SNE projections of the textures. Textures were described with the HANDCRAFTED feature set. Fig. 8 shows the projection of the Classical and Electronic genres of the ISMIR dataset.

A comparison between the spaces spanned by selecting 5 textures from each track with K-Means (Fig. 8(a)) and with linear downsampling (Fig. 8(b)) shows that the K-Means selection leads to a higher amount of observable clusters in the full-length datasets. This difference is not observable when 10 s excepts are used instead of full tracks, as shown in Figs. 8(c) and 8(d). When more textures are selected from each track, the number of clusters increases in both KMEANSC and LINSPACE, as shown in Figs. 8(e) and 8(f). Moreover, the number of clusters in KMEANSC seems higher than with LINSPACE. However, the difference in cluster numbers is much smaller than when 5 textures were used.

7. Discussion

The results in Section 6.1 indicate that as the number of textures per track increases, the results improve as well. This behaviour is seen in all texture selectors and applies to all feature sets and datasets. In a machine learning perspective this is expected due to the greater number of training examples. However, there were some cases using 5 textures that the results did not improve over the single-texture FTS baseline.

The visualisation of single tracks described in Section 6.4.1 showed that KMEANSC tends to select textures that are evenly distributed in feature space (Fig. 7(c)). Audio Sample 1 showed that this translates into diverse audio textures, capturing the sound variety within the track. In Section 6.4.2 we show that the feature space span of KMEANSC with 5 textures per track



(a) KNN (b) KNN+ANOVA

(c) SVM (d) SVM+ANOVA

Fig. A.10. Average F1-score across 10 RANDOM Folds for the GTZAN Dataset. Results for all texture selectors are shown. The colour bars show the results for 5, 20, and 40 selected textures in applicable selectors.

is greater than the span obtained by LINSPACE in full-length datasets (Figs. 8(a) and 8(b)). This can be a result of the greater sound variety captured by KMEANSC. This richer feature space obtained by KMEANSC was used to build the classifiers with 5 textures per track that were able to improve results over the FTS baseline, as shown in Tables 5a and 5b. The LINSPACE and RANDOM texture selectors, on the other hand, were unable to improve results.

When texture selection is performed in full-length tracks, as the ones in LMD and ISMIR, capturing sound diversity at random is challenging, specially with as few as 5 textures. LINSPACE and RANDOM do not pursue diversity, as they are content-agnostic. Thus, selecting diverse textures becomes a matter of chance. As heard in Audio Sample 2, this translated into sounds that are more common within the track. This is likely because common sounds occupy most of the textures. This makes it easier to randomly select common textures and to miss less common ones. On the other hand, KMEANSC optimises the centroids to describe different trends in texture data, which are translated into distinct sounds, as heard in Audio Sample 1.

As the track length gets shorter, it is more likely for it to become less varied. As such, capturing sound diversity at random becomes trivial as tracks get shorter and more homogeneous. Thus, it is more likely that the feature space span of different texture selectors become similar, as shown in Figs. 8(c) and 8(d). As a result, the classification performance using 5 textures per track of all texture selectors is statistically the same in 10 s datasets, as shown in Tables 5a and 5b. For the 30 s datasets, GTZAN-RANDOM and GTZAN-ARTF, the results are also similar among different texture selectors, although KMEANSC achieves greater improvements over the baseline.

As the number of selected textures per track increases, it is more likely that the resulting feature space span also increases. This can be seen by comparing Fig. 8(a) with 8(e) and Figs. 8(b) and 8(f). A greater span of the feature space can promote generalisation. This is verified by the results in Tables 5a and 5b, which shows that all results with 20 textures per track perform significantly better than with 5 textures in most datasets. In shorter datasets however, the improvements from increasing the number of textures are not as high. Tracks in shorter datasets are more likely to be homogeneous with respect to its texture variety. Therefore, fewer textures per track are able to describe all textural variety when compared to full-length datasets.

Texture selection allows downsampling datasets to reduce computing and storage requirements. A way of measuring this is comparing to the classification results when all textures are used for model training with the results of texture selection. In Section 6.1.2 we presented these results. The three texture selectors with 20 textures per track achieved results that were not statistically different than the ALL baseline in all datasets. This shows that it is possible to downsample datasets to just a fraction of its total textures and still achieve the same results. With the KNN classifier, KMEANSC was able to achieve significantly better results over using all textures in most cases. KNN performance is known to get degraded in the presence of outliers. When all textures are used, the probability of outliers is higher. As with KMEANSC we use the centroids as textures, the probability of selecting outliers within tracks is lower than with ALL, LINSPACE and RANDOM. Table 6b shows how in most datasets the results with KMEANSC and KNN were superior to the results with ALL (the baseline), LINSPACE and RANDOM. It also shows that most improvements over the ALL baseline were achieved by the KMEANSC texture selector.

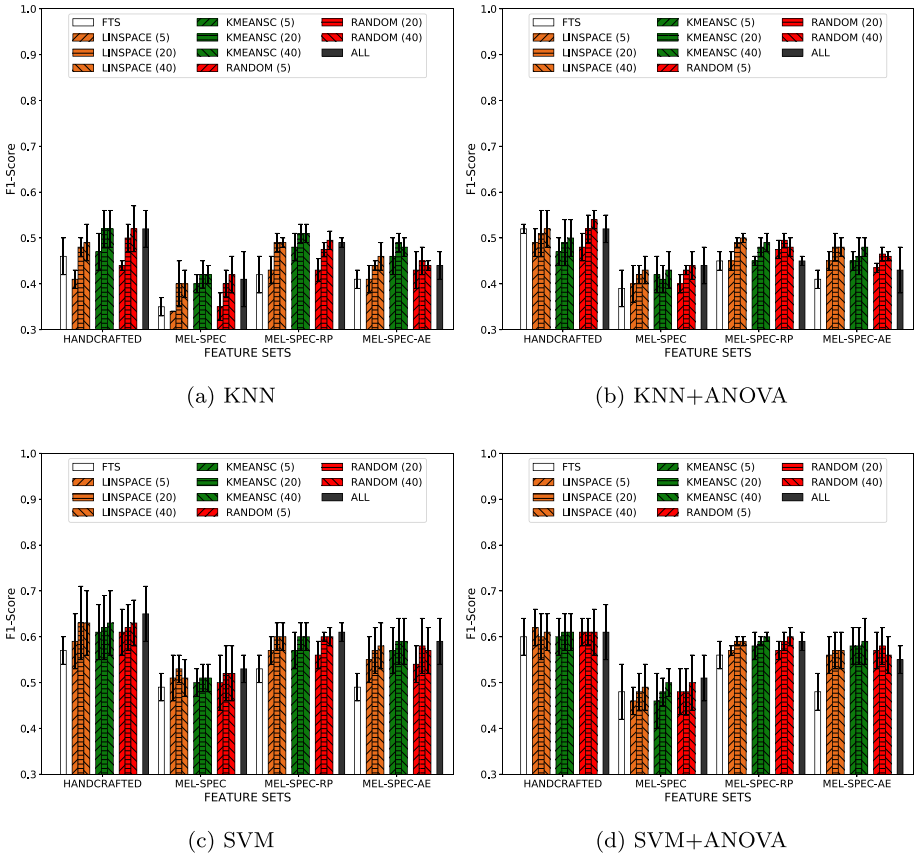


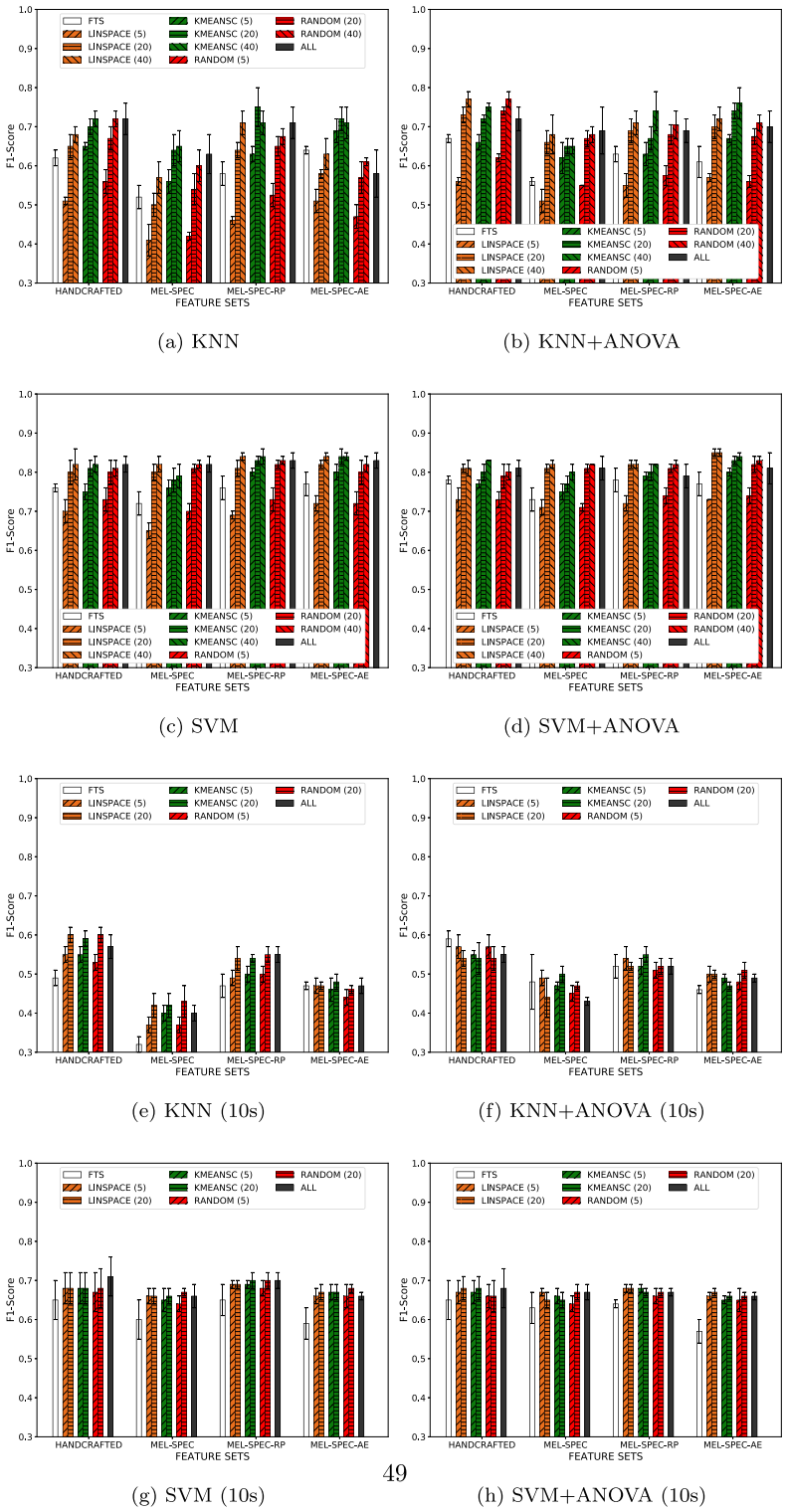
Fig. A.11. Average F1-score across 3 ARTF Folds for the GTZAN Dataset. Results for all texture selectors are shown. The colour bars show the results for 5, 20, and 40 selected textures in applicable selectors.

The K-Means algorithm usually starts picking random points as initial centroid guesses. To make sure that the K-Means optimisation improves classification performance over random selection, we also evaluated the RANDOM texture selector. The results presented in Section 6.1.1 shows that the improvements over the single-texture FTS baseline are larger with KMEANSC than with RANDOM. As discussed above, this is most likely due to KMEANSC selecting a greater variety of textures than RANDOM. This makes the feature space spanned by KMEANSC to describe a greater variety of sounds, promoting better generalisation.

Feature selection is often used in machine learning systems as a way to improve classification performance and speeding up model training. The results combining ANOVA feature selection and texture selection were presented in Section 3.4. The changes in results with feature selection in systems using SVM was not significant in most cases. In most cases with significant changes the results were improvements over the no selection baseline. This indicates that although most likely ANOVA feature selection does not improve results, it is harmless to use it along with SVM. By design SVM works well in high-dimensional feature spaces. Thus, removing features not linearly-correlated to labels does not impact the results so much. On the other hand, most results with KNN ANOVA changed significantly over the baseline. Most of the significant changes were positive. Feature selection lowers the dimensionality of the textures, thus alleviating the curse of dimensionality that has a detrimental effect on KNN. This can be the cause for the significant improvements seen in the KNN ANOVA experiments. This shows that most likely feature selection and texture selection have a complementary effect with KNN. FTS was the representation which feature selection had the least impact, with either SVM or KNN. On the other hand, the effect was much more positive in the representations based on multiple textures, mainly ALL and LINSPACE. This indicates that a representation based on multiple textures offers further opportunities for feature selection to improve classification performance.

8. Conclusions and future work

In this work we presented a study about the effect of texture selection on automatic music genre classification in a variety of machine learning setups. We presented a novel texture selection method based on K-Means clustering, aiming to select diverse textures from each track. We evaluated three texture selectors on four feature sets, as well a univariate feature selection strategy. To verify how texture selection works in a variety of data distributions, we evaluated four well-known and publicly available datasets. Our results show that texture selection is able to significantly improve over the single-texture FTS baseline, while achieving comparable results to using all textures to describe each track. We also showed that to improve performance, texture selection needs to capture texture variety within tracks. This leads to a greater span in feature space, as verified in the t-SNE visualisations, promoting generalisation. We also show that the proposed KMEANSC texture selection method is able to capture a diverse set of textures from each track, as shown in the PCA projections and corresponding audio samples. This is useful in datasets with full-length tracks, since capturing diversity at random becomes a challenge. As a result, our KMEANSC texture selector was able to achieve better results than the other selectors with 5 textures per track in full-length datasets. While KMEANSC with 5 textures per track was on average only 3% worse than using all textures, LINSPACE and RANDOM performed 10% and



(a) KNN (b) KNN+ANOVA

(c) SVM (d) SVM+ANOVA

(e) KNN (10s) (f) KNN+ANOVA (10s)

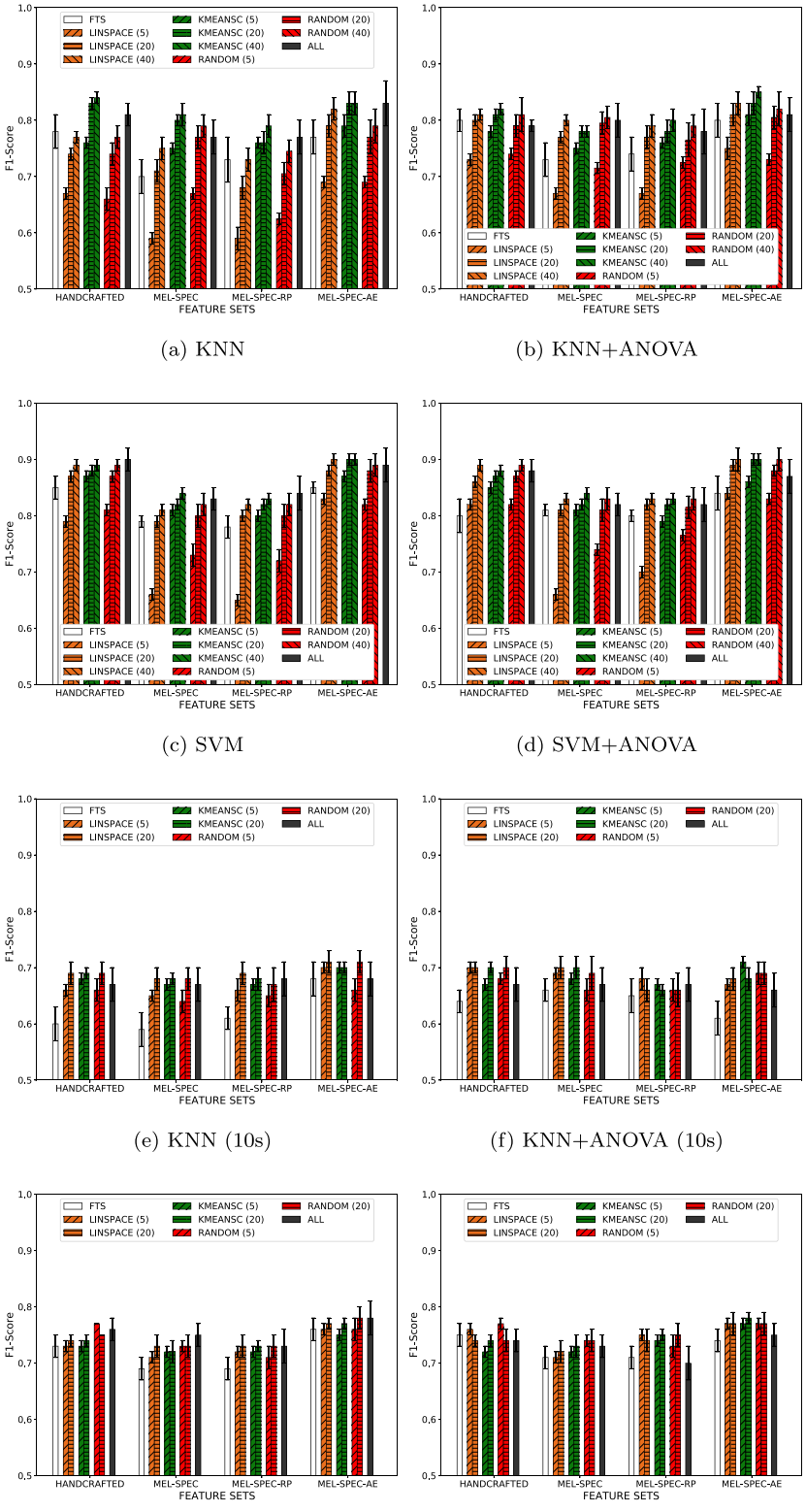
(g) SVM (10s) (h) SVM+ANOVA (10s)

Fig. A.12. Average F1-score across 3 Folds for the LMD Dataset. Results for all texture selectors are shown. The colour bars show the results for 5, 20, and 40 selected textures in applicable selectors.

10.6% worse, respectively. In shorter datasets, the results show that in general all texture selection datasets achieve comparable results.

A simplification we afforded in automatic music genre classification is that we consider a bag of frames approach where all textures have the same importance. However, this simplification is not appropriate in most problems. We want to investigate texture selection methods that can be used in cases where the importance of textures varies throughout the audio. This can be used in problems such as music and speech emotion recognition, and acoustic event detection.

For future works we want to evaluate further clustering methods for selecting textures within tracks. We are also working on further downsampling the selected textures by filtering textures



(a) KNN (b) KNN+ANOVA

(c) SVM (d) SVM+ANOVA

(e) KNN (10s) (f) KNN+ANOVA (10s)

(g) SVM (10s) (h) SVM+ANOVA (10s)

Fig. A.13. F1-scores for the ISMIR Dataset (ISMIR2004 contest train/test split). Results for all texture selectors are shown. The colour bars show the results for 5, 20, and 40 selected textures in applicable selectors.

of the same class according to information-gain criteria. Such a filtering scheme can be used as regularisation that can lead to a better compromise between bias and variance in the resulting machine learning model.

All Python code used in the experiments is available on GitHub.[3] The parameters we evaluated are described in Section 4.1.

[3] https://github.com/julianofoleiss/music_classification_framework.git

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106127.

Acknowledgements

The authors would like to thank the University of Campinas and the Universidade Tecnológica Federal do Paraná for supporting this research through the DINTER agreement. Thanks to Prof. Dr. Rogerio Aparecido Gonçalves for lending GPUs to support large-scale SVM training.

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A. KNN and SVM results for all datasets

See Figs. A.9–A.13.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.asoc.2020.106127.

References

[1] G. Tzanetakis, P. Cook, Musical genre classification of audio signals, IEEE Trans. Speech Audio Process. 10 (5) (2002) 293–302.
[2] J.-J. Aucouturier, B. Defreville, F. Pachet, The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music, J. Acoust. Soc. Am. 122 (2) (2007) 881–891.
[3] G. Marques, T. Langlois, F. Gouyon, M. Lopes, M. Sordo, Short-term feature space and music genre classification, J. New Music Res. 40 (2011) 127–137.
[4] C.M. Yeh, L. Su, Y. Yang, Dual-layer bag-of-frames model for music genre classification, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 246–250.
[5] Y.M. Costa, L.S. Oliveira, C.N. Silla, An evaluation of convolutional neural networks for music classification using spectrograms, Appl. Soft Comput. 52 (2017) 28–38.
[6] T. Kim, J. Lee, J. Nam, Comparison and analysis of samplecnn architectures for audio classification, IEEE J. Sel. Top. Sign. Proces. 13 (2) (2019) 285–297.
[7] H. Yang, W.-Q. Zhang, Music genre classification using duplicated convolutional layers in neural networks, in: INTERSPEECH 2019, 2019.
[8] C. Senac, T. Pellegrini, F. Mouret, J. Pinquier, Music feature maps with convolutional neural networks for music genre classification, in: Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing, CBMI '17, ACM, New York, NY, USA, 2017, pp. 19:1–19:5.
[9] J. Salamon, J.P. Bello, Deep convolutional neural networks and data augmentation for environmental sound classification, IEEE Signal Process. Lett. 24 (3) (2017) 279–283.
[10] R.L. Aguiar, Y.M.G. Costa, C.N. Silla, Exploring data augmentation to improve music genre classification with convnets, in: 2018 International Joint Conference on Neural Networks, IJCNN, 2018, pp. 1–8.
[11] S. Shin, H. Yun, W. Jang, H. Park, Extraction of acoustic features based on auditory spike code and its application to music genre classification, IET Signal Process. 13 (2) (2019) 230–234.
[12] Y. Yu, S. Luo, S. Liu, H. Qiao, Y. Liu, L. Feng, Deep attention based music genre classification, Neurocomputing.
[13] T. Kobayashi, A. Kubota, Y. Suzuki, Audio feature extraction based on sub-band signal correlations for music genre classification, in: 2018 IEEE International Symposium on Multimedia, ISM, 2018, pp. 180–181.
[14] S. Oramas, O. Nieto, F. Barbieri, X. Serra, Multi-label music genre classification from audio, text, and images using deep features, in: Proceedings of the 18th ISMIR Conference, 2017.
[15] J.-J. Aucouturier, F. Pachet, Improving timbre similarity: How high's the sky? J. Negat. Results Speech Audio Sci..
[16] E. Pampalk, A. Flexer, G. Widmer, Improvements of audio-based music similarity and genre classificaton, in: Proceedings of the 6th International Conference on Music Information Retrieval, 2005.
[17] J. Macqueen, Some methods for classification and analysis of multivariate observations, in: 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.
[18] J. Lee, J. Nam, Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging, IEEE Signal Process. Lett. 24 (8) (2017) 1208–1212.
[19] K. Choi, G. Fazekas, M. Sandler, K. and Cho, Transfer learning for music classification and regression tasks, in: Proceedings of the 18th ISMIR Conference, 2017.
[20] S. Dieleman, B. Schrauwen, End-to-end learning for music audio, in: Acoustics, Speech and Signal Processing, ICASSP, 2014 IEEE International Conference on, IEEE, 2014, pp. 6964–6968.
[21] M.A. Davenport, M.F. Duarte, M.B. Wakin, J.N. Laska, D. Takhar, K.F. Kelly, R.G. Baraniuk, The smashed filter for compressive classification and target recognition, in: Computational Imaging V, Vol. 6498, International Society for Optics and Photonics, 2007, p. 64980H.
[22] R.G. Baraniuk, V. Cevher, M.B. Wakin, Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective, Proc. IEEE 98 (6) (2010) 959–971.
[23] W.B. Johnson, J. Lindenstrauss, Extensions of Lipschitz mappings into a Hilbert space, Contemp. Math. (26) (1984) 189–206.
[24] Y. Panagakis, C. Kotropoulos, G.R. Arce, Music genre classification via sparse representations of auditory temporal modulations, in: 2009 17th European Signal Processing Conference, 2009, pp. 1–5.
[25] K.K. Chang, J.-S.R. Jang, C.S. Iliopoulos, Music genre classification via compressive sampling, in: Proceedings of the 11th ISMIR Conference, 2010.
[26] M. Banitalebi-Dehkordi, A. Banitalebi-Dehkordi, Music genre classification using spectral analysis and sparse representation of the signals, J. Signal Process. Syst. 74 (2) (2014) 273–280.
[27] S. Dubnov, Generalization of spectral flatness measure for non-gaussian linear processes, IEEE Signal Process. Lett. 11 (8) (2004) 698–701.
[28] M. Hunt, M. Lennig, P. Mermelstein, Experiments in syllable-based recognition of continuous speech, ICASSP '80, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 5, 1980, pp. 880–883.
[29] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.
[30] B.K. Baniya, J. Lee, Z. Li, Audio feature reduction and analysis for automatic music genre classification, in: 2014 IEEE International Conference on Systems, Man, and Cybernetics, SMC, 2014, pp. 457–462.
[31] Z. Wen, J. Shi, Q. Li, B. He, J. Chen, ThunderSVM: A fast SVM library on GPUs and CPUs, J. Mach. Learn. Res. 19 (2018) 1–5.
[32] V. Vapnik, Statistical Learning Theory, Wiley, 1998.
[33] The International Society for Music Information Retrieval (ISMIR), ISMIR2004 audio description contest-genre/artist id classification and artist similarity, 2004, http://ismir2004.ismir.net-/genrecontest.
[34] C. Silla, A.L. Koerich, C.A.A. Kaestner, The Latin music database, in: Proceedings of the 9th International Conference on Music Information Retrieval, Philadelphia, PA, USA, 2008.
[35] H. Homburg, I. Mierswa, B. Möller, K. Morik, M. Wurst, A benchmark dataset for audio classification and clustering. in: 6th Proceedings of the International Conference on Music Information Retrieval, 2005.
[36] B.L. Sturm, The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use, arXiv preprint arXiv:1306.1461.
[37] A. Flexer, A closer look on artist filters for musical genre classification. in: Proceedings of the 8th International Conference on Music Information Retrieval, 2007, pp. 341–344.
[38] B.L. Sturm, A survey of evaluation in music genre recognition, in: International Workshop on Adaptive Multimedia Retrieval, Springer, 2012, pp. 29–66.
[39] L. Nanni, Y.M. Costa, A. Lumini, M.Y. Kim, S.R. Baek, Combining visual and acoustic features for music genre classification, Expert Syst. Appl. 45 (2016) 108–117.