6

AUDIO BIGRAMS

6.1 INTRODUCTION

In chapter 2 and chapter 5, we argued that alignment algorithms are not a good solution for large-scale cover song retrieval: they are slow, and a database query based on pairwise comparison of songs takes an amount of time proportional the size of the database.

Starting from this observation, and inspired by audio fingerprinting research, we proposed in section 2.2.3 the umbrella MIR task of 'soft audio fingerprinting', which includes any scalable approach to the content-based identification of music documents. Soft audio fingerprinting algorithms convert fragments of musical audio to one or more fixed-size vectors that can be used in distance computation and indexing, not just for traditional audio fingerprinting applications, but also for retrieval of cover songs and other variations from a large collection. We reviewed the most important systems that belong to these categories. Unfortunately, for the task of cover song detection, none of the systems have reached performance numbers close to those of the alignment-based systems.

This chapter provides a new perspective on soft audio fingerprinting. Section 6.2 presents a birds-eye view of research done on the topic. Next, section 6.3 identifies and formalizes an underlying paradigm that allows to see several of the proposed solutions as variations of the same model. From these observations follows a general approach to pitch description, which we will refer to as *audio bigrams*, that has the potential to produce both very complex and very interpretable

musical representations. Finally, section 6.4 present PYTCH, a Python implementation of the model that accommodates several of the reviewed algorithms and allows for a variety of applications, alongside an example experiment that illustrates its use. The toolbox can be used for the comparison of fingerprints, and for song and corpus description in general. It is available online and open to extensions and contributions.

6.2 SOFT AUDIO FINGERPRINTING

This section reviews the various music representations and transformations on which the most important soft audio fingerprinting techniques are based. The systems can be classified into four main groups: spectrogram-based approaches, constant-Q-based approaches, chroma-based approaches, and approaches based on melody.

Spectrogram-based Fingerprinting

Wang and Smith's seminal 'landmark-based' strategy, reviewed in section 2.2.3 is centered on the indexing of pairs of peaks in the spectrogram. In other words: the representation used by the system is the spectrogram, the transformations that are applied are peak detection and pairing of the resulting vectors into a length-4 array.

Constant-Q-based Fingerprinting

At least three systems in the literature have successfully applied the same idea to spectrograms for which the frequency axis is divided into logarithmic rather than linearly spaced bins [51, 181, 192]. Such spectral representations are generally referred to as constant-Q transforms (section 2.1.1) All three systems aim to produce fingerprints that are robust to pitch shifting, an effect that is often applied by DJ's at radio stations and in clubs. The system by Van Balen specifically aims to identify cases of sampling, a compositional practice in which pieces of recorded music are transformed and reused in a new work.

In each case, the idea is that the logarithmic, constant-Q spectrogram preserves relative pitch (also logarithmic in frequency). Transformations used are again peak-detection, and the joining of peaks into pairs [51, 192] and triplets [181].

Chroma-based Fingerprints

The first to perform a truly large-scale cover song retrieval experiment, Bertin-Mahieux and others have taken the landmarks idea and applied it to chroma representations of the audio. Peaks in the beat-aligned chroma features are again paired, into 'jumpcodes'. In an evaluation using the very large *Million Song Dataset* (MSD, see section 3.3.2), the approach was found to be relatively unsuccessful [12, 13].

A follow-up study by Bertin-Mahieux also uses chroma features, but follows an entirely different approach. Beat-aligned chroma features are transformed using the two-dimensional discrete Fourier transform (DFT), to obtain a compact summary of the song that is somewhat hard to interpret, but roughly encodes recurrence of pitch intervals in the time domain [13]. Since only the magnitudes of the Fourier coefficients are used, this '2DFTM' approach is robust to both pitch shifting and tempo changes by design. Results are modest, but formed the state-of-the art in scalable cover song retrieval at the time.

Humphrey et al. have taken this idea further by applying a number of feature learning and dimensionality reduction techniques to the above descriptor, aiming to construct a sparse geometric representation that is more robust against the typical variations found in cover songs [75]. The method performs an initial dimensionality reduction on the 2DFTM features, and then uses the resulting vectors to learn a large dictionary (using k-means clustering), to be used as a basis for a higher-dimensional but sparser representation. Finally, supervised Linear Discriminant Analysis (LDA) is used to learn a reduced embedding optimized for cover song similarity. Their method achieves an increase in precision, but not in recall.

Other soft audio fingerprinting systems are simpler in concept, e.g., the procedure proposed by Kim et al. [90]. This system takes 12-

dimensional chroma and delta chroma features and computes their 12×12 covariance matrices to obtain a global fingerprint for each song. This effectively measures the co-occurrence of pitch energies and pitch onsets, respectively. This relatively simple strategy achieves good results on a small test set of classical music pieces. A later extension by the same authors introduces the more sophisticated 'dynamic chroma feature vectors', which roughly describe pitch intervals [89].

A very similar family of fingerprints was proposed in chapter 5. The chroma correlation coefficients (equation 40) and harmonic interval co-occurrence (equation 48) descriptors follow a covariance-like transformation to compute co-occurrence.

Melody-based Fingerprints

A second set of fingerprint functions proposed in chapter 5 also makes use of melody information, as extracted by a melody estimation algorithm: the pitch bihistgram (equation 39), melodic interval bigrams (equation 46), and the harmonization and harmonization intervals descriptors (equations 43, 49).

The first two aim to measure ordered co-occurrence in melodic pitch—counting co-occurrence of pitch class activations given a certain maximum offset in time. The latter describe the co-occurrence of harmonic and melodic pitch.

6.3 UNIFYING MODEL

6.3.1 *Fingerprints as Audio Bigrams*

The fingerprinting model we propose in this chapter builds on the following observation:

> Many of the fingerprinting methods listed in the previous section detect salient events in a time series, and pair these over one or more time scales to obtain a set or distribution of bigram-like tokens.

The paradigm identified here will be referred to as the *audio bigram* paradigm of fingerprinting. We propose the following definition:

> Audio bigrams are pairs of salient events, extracted from a multidimensional time series, that co-occurr within a specified set of timescales.

Audio bigrams and distributions over audio bigrams can be used as fingerprints.

This definition will be further formalized in the next section. However, it may already become apparent how some of the example systems from section 6.2 can be mapped to the above formulation of the audio bigrams model.

In Wang's landmark approach [199], for example, the salient events are peaks in the linear spectrum (X = DFT magnitudes), and the time scales for pairing are a set of fixed, linearly spaced offsets τ up to a maximum horizon Δt (τ and Δt in frames).

$$\tau = \{1, 2 \ldots \Delta t\} \tag{51}$$

yielding a set of Δt peak bigram distributions for the total of the fragment.

The constant-Q-based variants [51,192] and the jumpcodes approach [12] reviewed above are precisely analogous, with salient events as peaks in the logarithmic spectrum (X = CQT) and beat-aligned chroma features, respectively.

For the case of the melody bigrams, the salient events are pitch class activations pertaining to the melody and the time scale for pairing is a single range of offsets

$$\tau \leq \Delta t. \tag{52}$$

Chroma (and delta chroma) covariance and correlation matrix features are even simpler under this paradigm: pitch activations are only paired if they are simultaneous, i.e. $\tau = 0$.

Two approaches that don't seem to be accommodated at first sight, are Bertin-Mahieux and Humphrey's algorithms based on the 2D DFT. In the remainder of this section, we will show that:

1. a formulation of the audio bigram model exists that has the additional advantage of easily being vectorized for efficient computation,
2. the vectorized model is conceptually similar to the 2D Fourier transform approach to cover song fingerprinting,
3. the model is closely related to convolutional neural network architectures and can be used for feature learning.

It is good to point out that the model will not accommodate all of the algorithms completely. Notably, in the adaptation of landmark-based fingerprinting as described here, some of the time information of the landmarks is lost, namely, the start time of the landmarks. We believe this can ultimately be addressed,[1] but currently don't foresee any such adaptation, as the primary aim at this stage is to explore and evaluate the commonalities between the algorithms.

6.3.2 *Efficient Computation*

In this section, we further formalize the model and characterize its computational properties by proposing an efficient reformulation. The first step to be examined is the detection of salient events.

Salient Event Detection

In its simplest form, we may see this component as a peak detection operation. Peak detection in a 2-dimensional array or matrix is most simply defined as the transformation that marks a pixel or matrix cell as a peak (setting it, say, to 1) if its value is greater than any of the values in its immediate surroundings, and non-peak if it's not (setting it to 0).

Peak detection may be vectorized using dilation. Dilation, denoted with $\oplus$, is an operation from image processing, in which the value of

1 e.g. by not extracting one global fingerprint, but fingerprinting several overlapping segments and pooling the result, cf. [13,75]

a pixel in an image or cell in a matrix is set to the maximum of its surrounding values. Which cells or pixels constitute the surroundings is specified by a small binary 'structuring element' or masking structure.

Given a masking structure S_m, the dilation of the input matrix X with S_m is written as $S_m \oplus X$. The peaks, then, P are those positions in X where $X \geq S_m \oplus X$. In matrix terms:

$$P = h(X - S_m \oplus X) \tag{53}$$

where

$$h(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{54}$$

the Heaviside (step) function.

As often in image processing, convolution, denoted with $\otimes$, can be used as well. We get:

$$P = h(X - S_m \otimes X) \tag{55}$$

where $h(x)$ as above, or if we wish to retain the peak intensities,

$$h(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{56}$$

the rectification function.

Equivalently, we may write

$$P = h(S \otimes X) \tag{57}$$

where S is a mostly negative kernel with center 1 and all other values equal to $-S_m$, similar to kernels used for edge detection in images (top left in figure 26).

The latter approach, based on convolution, allows for the detection of salient events beyond simple peaks in the time series. As in image processing and pattern detection elsewhere, convolutional kernels can be used to detect a vast array of very specific patterns and structures ranging, for this model, from specified intervals (e.g. fifths or sevenths) over major and minor triads to 'delta-chroma' and particular interval jumps. See figure 26 for simplified examples.

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ 17 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 2 & -1 & -1 & -1 \\ 2 & -1 & 2 & -1 & -1 & -1 \\ 2 & -1 & 2 & -1 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 5 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 5 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 26.: Examples of event-detecting kernels. Rows are time frames, columns can be thought of as pitch classes or frequency bins. They roughly detect, respectively, edges or single peaks, a two-semitone interval sounding together, a two-semitone jump, and 'delta chroma'.

The above is, as is said, a 'vectorized' algorithm, meaning that it is entirely formulated in terms of arrays and matrices. When all computations are performed on arrays and matrices, there is no need for explicit *for* loops. For practical implementations, this means that, in high-level programming languages like Matlab and Python/Numpy, code will run significantly faster, as the *for* loops implied in the matrix operations are executed as part of dedicated, optimized, lower-level code (e.g., C code). Additionally, convolution can be executed very efficiently using FFT.

Co-occurrence Detection

Co-occurrence can be formalized in a similarly vectorized way. Consider that the correlation matrix of a multidimensional feature can be written as a matrix product:

$$F = P^\top \cdot P \tag{58}$$

provided each column in P has been normalized by subtracting of the mean and dividing by the standard deviation. (When only the subtraction is performed, F is the covariance matrix). If P is a chroma feature, for example, the resulting fingerprint measures the co-occurrence of harmonic pitch classes.

When a certain time window for pairing needs to be allowed, things get a little more complicated. We propose an efficient approach in which dilation or convolution is again applied prior to the matrix multiplication. In this case, the structuring element we need is a binary column matrix (size along the pitch dimension is one) of length $2\Delta t + 1$, denoted T.

$$T = [\underbrace{0 \quad \dots \quad 0}_{\Delta t} \quad 0 \quad \underbrace{1 \quad \dots \quad 1}_{\Delta t}]^\top \tag{59}$$

The co-occurrence feature can then be defined as

$$F = P^\top \cdot (T \oplus P). \tag{60}$$

where P may contain, for example, the melody, $M(t, p)$, a chroma-like matrix containing 1 when a pitch class p is present at time t in the melody and 0 everywhere else.

To see how the above F is mathematically equivalent to the proposed co-occurrence matrix, consider

$$P' = (T \oplus P). \tag{61}$$

By definition of $\oplus$,

$$P'(t, i) = \max_\tau (P(t + \tau, i)) \quad \tau = 1 \dots \Delta t \tag{62}$$

so that

$$F(i, j) = \sum_t P(t, i) \ \max_\tau (P(t + \tau, j)) \quad \tau = 1 \dots \Delta t \tag{63}$$

which, for a binary melody matrix $M(t, p)$, translates to

$$B(p_1, p_2) = \sum_t M(t, p_1) \ \max_\tau (M(t + \tau, p_2)) \tag{64}$$

the definition of the pitch bihistogram as given in section 5.2.1 (equation 39).

Assuming that the melody matrix M is based on the melody $m(t)$, this also translates to

$$F(i, j) = \sum_t \max_\tau \begin{cases} 1 & \text{if } m(t) = i \text{ and } m(t + \tau) = j, \\ 0 & \text{otherwise,} \end{cases} \tag{65}$$

the standard definition of the co-occurrence matrix over discrete one-dimensional data.

Alternatively, convolution can be applied, and we get

$$F = P^\top \cdot (T \otimes P) \tag{66}$$

or, in terms of $m(t)$,

$$F(i,j) = \sum_t \sum_\tau \begin{cases} 1 & \text{if } m(t) = i \text{ and } m(t+\tau) = j, \\ 0 & \text{otherwise,} \end{cases} \tag{67}$$

provided S is again binary.

The difference between these two types of co-occurrence matrix is small for sufficiently sparse M, in which case $\max_\tau \approx \sum_\tau$. This is generally true for natural language data, a context in which co-occurrence is often used. It also holds for the sparse peak constellations used in classic landmark-based audio fingerprinting. For more general, dense matrices, the convolution-based F will scale with the density of M while the dilation-based F will not. When efficiency is important, the convolution-based approach should be preferred, so there is an advantage in enforcing sparsity in P.

Methods such as landmark extraction, that do *not* sum together all co-occurrences for all offsets $\tau \leq \Delta t$, can be be implemented using multiple T_k, each of the form:

$$T_k = [\underbrace{0 \quad \dots \quad 0}_{k} \quad 0 \quad \underbrace{0 \quad \dots \quad 0 \quad 1}_{k}]^\top \tag{68}$$

These then yield a set of F_k, one for each offset $k = 1, 2 \dots K$. Naturally, more complex T_k are possible as well.

We conclude that the pairing of salient events over different time scales can be completely vectorized for efficient computation using image processing techniques such as dilation, convolution or both.

Summary

Combining the above gives us a simple two-stage algorithm for the extraction of audio bigram distributions.

Given an input times series X (time frames as rows), a set of n masking structures $\{S_i\}$ and a set of K structural elements $\{T_k\}$ specifying the time scale for co-occurrence, we apply

1. *salient event detection* using
   - convolution with S_i:

$$X'_i = S_i \otimes X \tag{69}$$

   - rectification:

$$P(i,t) = h(X'_i(t)) \tag{70}$$

   $i = 1 \ldots n$.

2. *co-occurrence detection* using
   - convolution with T_k:

$$F_k(i,j) = P^\top \cdot (T_k \otimes P) \tag{71}$$

   $i, j = 1 \ldots n$ and $k = 1 \ldots K$.
   - optional normalization.

so that $F_k(i,j)$ in the matrix form of fingerprint F_k encodes the total amount of co-occurrences of S_i and S_j over the time scale specified by T_k.[2]

6.3.3 *Audio Bigrams and 2DFTM*

We have already shown how pitch bihistogram, chroma covariance and chroma correlation coefficients can be implemented using the above algorithm. Implementing delta chroma covariance, as in [90],

2 The above example assumes convolutions are used, and the further restriction that all S_i have a number of columns equal to that of X, so that each convolution yields a one-dimensional result. Cases where the number of rows in S is smaller are equivalent to using a set S_i of i shifted filters, where i is the difference in number of columns between S and X.

is just as easy: the difference filter shown in figure 26 (bottom right) is applied to the chroma before computing co-occurrence, everything else is the same. Spectrogram, constant-Q and chroma landmarks are also straightforward to implement: S is a peak detection kernel and $T_k = e_k, k = 1 \ldots K$.

Can this formalization also be linked to the 2DFTM and feature learning approaches by Bertin-Mahieux et al. and Humphrey et al.? The most straightforward intuition behind the output of the 2D Fourier magnitude coefficients over a patch of chroma features, is that it encodes periodic patterns of pitch class activation in time.

The model proposed in this chapter measures co-occurrences of events given a set of timescales. In other words, its aspires to do just the same as the 2DFTM-based systems, but with a constraint on what kinds of recurrence in time and pitch space are allowed, by linking it to the bigram paradigm that has been successful in other strands of audio fingerprinting.

Audio Bigrams and Convolutional Neural Networks

The above set of transforms is very similar to the architecture of convolutional neural networks as used in computer vision and artificial intelligence.

Convolutional neural networks (CNN) are a class of artificial neural networks in which a cascade of convolutional filters and non-linear activation functions is applied to an input vector or matrix (e.g. an image). Common non-linear functions include sigmoid functions (e.g. tanh) and the rectification function, used in so-called rectified linear units or ReLU's.

CNN are much like other neural networks, in that most of the layers can be expressed as a linear operation on the previous layer's output followed by a simple non-linear scaling of the result. The coefficients of these linear operations can be seen as the 'weights' of connections between neurons, and make up the majority of the network's parameters. These parameters can typically be learned given a large dataset of examples. An important advantage of CNN over other neural net-

works is that the connections are relatively sparse, and many of the weights are shared between connections, both of which make learning easier. However, learning these parameters in the context of variable-length time series presents an extra challenge: the dimensions of the output and the the number of weights of a typical neural network are assumed to be constant.

The audio bigram model, as it is summarized in section 6.3.2, only consists of convolutions, one non-linear activation function h and a dot product. This makes it a type of convolutional neural network, and a rather simple one: there are relatively few parameters. Conveniently, and to our knowledge, unprecedentedly, the audio bigram approach circumvents the variable-length input issue by exploiting the fixed size of the dot product in Equation 71. The correspondence between audio bigrams and CNN's suggests that, for a given task, optimal matrices S_i and T_k may perhaps be learned from a sufficiently large collection of matches and mismatches.

Audio Bigrams and 2DFTM

Finally, because of the convolution-multiplication duality of the DFT, the audio bigram model can be considered the non-Fourier domain analogue of the k-NN-based system proposed by Humphrey, who describes their system as 'effectively performing convolutional sparse coding' [75].

Future experiments will determine whether standard learning algorithms using back-propagation can be used for this kind of convolutional architecture, and whether an improvement in tasks like sample identification and cover song detection can be achieved using the resulting models.[3]

Figure 27.: Classes (Song, Experiment) and modules (fingerprinting, plotting) in the soft audio fingerprinting toolbox PYTCH.

6.4 IMPLEMENTATION

6.4.1 PYTCH

We provide an implementation of the above ideas in the form of PYTCH, a Python toolbox for soft audio fingerprinting and pitch-based song description available at `www.github.com/jvbalen/pytch`.

The toolbox builds on two primary classes and one important module. A class *Song* contains an ID, a clique ID (denoting the group of covers it belongs to), and any available raw features (e.g., chroma, melody). Centrally in the toolbox is the *fingerprint* module, containing the fingerprinting transforms. It implements a set of fingerprinting methods using some of the feature transforms reviewed out in section 6.2 in the implementation proposed in 6.3. New transforms and new configurations of the existing architecture can be added here. On top of this, a class *Experiment* can be used to evaluate fingerprinting methods, and a *plotting* module can be used for visualization. Figure 27 illustrates the class and module structure of the toolbox.

3 Experiments will be documented at `https://github.com/jvbalen/cover_id`

6.4.2 *Code Example*

In the following (Python) example, an experiment is run on a set of 100 candidates and queries for which the *Song* class has access to a file with chroma base features. This involves three steps.

- Songs and their features are loaded upon construction of the *Collection* objects `queries` and `candidates`.

```
import song

query_ids = range(0,100)
candidate_ids = range(100,200)

file_dir = 'test_corpus\'
file_list = 'list.txt'

queries = [song.Song(file_dir, file_list, id)
           for id in query_ids]
candidates = [song.Song(file_dir, file_list, id)
              for id in candidate_ids]
```

- A fingerprinting function and its parameters are chosen and passed to the object `my_experiment` of the *Experiment* class.

```
import fingerprint as fp
import experiment as xp

fingerprint_type = fp.pitch_bihistogram
fingerprint_params = {'win': 0.5,
                      'normfunction': 'whiten',
                      'keyhandling': 'transpose'}
my_experiment = xp.Experiment(queries, candidates,
                              fingerprint_type,
                              fingerprint_params)
```

- Finally, the experiment is run. The experiment is small enough for the system to compute all pairwise (cosine) distances between the fingerprints.

```
my_experiment.run(dist_metric='cosine',
                  eval_metrics=['map'])
print my_experiment.results
```

In most practical uses of this toolbox, it is advised to set up a new module for each new dataset, overriding the *Song* and *Experiment* constructors to point to the right files given an ID. In the future, an index may be added to store and retrieve fingerprint hashes for large collections.

6.4.3 *Example Experiment*

We now demonstrate in an example experiment how bigram-based fingerprints can be compared, by testing a number of configurations of the system in a cover song retrieval experiment.

As a dataset, we use a subset of the *Second Hand Song* dataset of 1470 cover songs.[4] The subset contains 412 'cover groups' or *cliques*, and for each of these we have arbitrarily selected one song to be the query. The other 1058 songs constitute the candidate collection. Since the subset is based on the second hand song dataset, we have access to pre-extracted chroma features provided by the Echo Nest. Though not ideal, as we don't know exactly how these features were computed (see section 3.4), they make a rather practical test bed for higher-level feature development.

We implemented four bigram-based fingerprints: three kinds of chroma co-occurrence matrices (correlation, covariance, and chroma difference covariance following [90, 191]), and one chroma landmark system, roughly following [12]. The results, with a description of the kernels S and T, are given in Table 3. The chroma landmark strategy was optimized over a small number of parameters: T_k was settled on

4 http://labrosa.ee.columbia.edu/millionsong/secondhand

System	MAP	R_1	R_5
Random baseline	.012	.001	.002
Chroma correlation no S, no T	.181	.097	.155
Chroma covariance no S, no T	.223	.112	.194
Chroma difference covariance $S = [-1, 1]^\top$, no T	.114	.051	.107
Chroma landmarks S simple peak detection T_k of form $[\ldots 0, 1]^\top$	.367	.189	.340

Table 3.: Results table for the example cover song experiment. Chroma landmarks outperform other bigram-type fingerprints.

a set of length-k arrays where $k = 1 \ldots 16$. The best length of the peak-detecting matrix S for the system was found to be 32. Only one peak detection matrix was used.

As can be seen from the table, the chroma landmark system outperforms the other systems. We believe this supports the hypothesis that, starting from the kernels S and T that describe this transform, a more powerful representation can be learned.

6.5 CONCLUSIONS AND FUTURE WORK

We have reviewed a selection of soft audio fingerprinting methods, and described a fingerprinting model that allows to see these methods as variations of the *audio bigram* paradigm. The audio bigram model measures co-occurrence of pre-specified salient events in a multidimensional time series. We have presented an exploration of the computational architecture of the model and showed that it can be implemented as a particular type of convolutional neural network. The

model can therefore be optimised for specific retrieval tasks using supervised learning. Finally, we have introduced an implementation of the model, PYTCH.

As future work, we plan a more extensive evaluation of some of the existing algorithms the system is capable of approximating. Standard datasets like the *covers80* dataset can be used to compare results to existing benchmarks. If the results are close to what the original authors have found, PYTCH may be used to do a comparative evaluation that may include some variants of the model that have not previously been proposed.

We also intend to study the extent to which the convolutional network implementation of the model can be trained, and what kind of variants of the models this would produce. This can be done most easily using the complete *Second Hand Song* dataset, because a rather large number of train and test data will be required.