EURASIP Journal on Audio, Speech, and Music Processing

EMPIRICAL RESEARCH **Open Access**

AAM: a dataset of Artificial Audio Multitracks for diverse music information retrieval tasks

Fabian Ostermann[1*], Igor Vatolkin[1] and Martin Ebeling[2]

Abstract

We present a new dataset of 3000 artificial music tracks with rich annotations based on real instrument samples and generated by algorithmic composition with respect to music theory. Our collection provides ground truth onset information and has several advantages compared to many available datasets. It can be used to compare and optimize algorithms for various music information retrieval tasks like music segmentation, instrument recognition, source separation, onset detection, key and chord recognition, or tempo estimation. As the audio is perfectly aligned to original MIDIs, all annotations (onsets, pitches, instruments, keys, tempos, chords, beats, and segment boundaries) are absolutely precise. Because of that, specific scenarios can be addressed, for instance, detection of segment boundaries with instrument and key change only, or onset detection only in tracks with drums and slow tempo. This allows for the exhaustive evaluation and identification of individual weak points of algorithms. In contrast to datasets with commercial music, all audio tracks are freely available, allowing for extraction of own audio features. All music pieces are stored as single instrument audio tracks and a mix track, so that different augmentations and DSP effects can be applied to extend training sets and create individual mixes, e.g., for deep neural networks. In three case studies, we show how different algorithms and neural network models can be analyzed and compared for music segmentation, instrument recognition, and onset detection. In future, the dataset can be easily extended under consideration of specific demands to the composition process.

Keywords Artificial music dataset, Multitrack audio mixes, Algorithmic composition, Music segmentation, Instrument recognition, Source separation, Onset detection, Tempo estimation, Chord detection

1 Introduction

Annotated datasets are required to evaluate, compare, and optimize algorithms for various supervised music classification, regression, and event detection tasks: recognition of instruments and vocals, music segmentation, onset and chord detection, tempo and key estimation, and many more. Lots of datasets with annotations were introduced in recent years and decades, however, often with limitations.

First of all, many annotations focus on only one or few aspects, because most datasets have been created for an individual music information retrieval (MIR) task like tempo estimation or music segmentation. For instance, it is not possible to evaluate instrument and harmony recognition methods, when only onsets and segment boundaries are annotated. Second, for many reasons the annotations are not always precise. Sometimes, they are created by non-experts, and even expert annotations may be ambiguous. Third, the annotations seldom contain detailed reasoning for provided labels. For example, annotated segment boundaries usually do not describe directly whether these boundaries are related to timbral, harmonic, temporal, or rhythmic properties, which makes it hard to exhaustively evaluate the algorithms and to identify their individual weak

*Correspondence:
Fabian Ostermann
fabian.ostermann@tu-dortmund.de
[1] Department of Computer Science, TU Dortmund University, Dortmund, Germany
[2] Department of Arts Science, Institute of Musicology, TU Dortmund University, Dortmund, Germany

points; for such scenarios, "synthetic data generation allows you to address the problem in a more targeted way" [1]. Fourth, some datasets contain only lists of commercial music tracks which are not publicly available. Even if audio previews or previously extracted features are released with these datasets, it is not possible to extract further features for complete music pieces. Fifth, only seldom audio multitracks, which contain all single instrument tracks from the mix, are provided. In particular for deep neural networks, which benefit from large and augmented training sets, multitracks make it possible to develop more robust algorithms, because individual augmentation methods like loudness change or digital audio effect processing can be separately applied on different instrument tracks to create individual mixes with the same annotations. Finally, the acquiring and labeling of new data is typically a costly, exhausting, and error-prone procedure that may include the purchase of commercial music tracks and manual efforts on annotating ground truth events. In contrast, our dataset can easily be extended under consideration of specific music properties, e.g., increasing the number of simultaneously playing instruments, allowing non-harmonic tones, or introducing non-Western scales, in order to perfectly match desired use cases.

Our new dataset AAM[1] (Artificial Audio Multitracks) addresses all aforementioned issues and is designed for the evaluation of various MIR tasks: music segmentation and structure analysis, instrument recognition and source separation, onset detection, key and chord recognition, or tempo estimation. AAM contains both audio and symbolic data, which are perfectly aligned.

However, it is important to note that some applications like genre recognition cannot be performed with AAM, because all music pieces were algorithmically generated and, therefore, do not represent "real-world" human compositions. The dataset is not fully competitive to them and will probably not generalize well (at this first stage of development). Nevertheless, its great strength is that it allows for an exhaustive testing, individual optimization, and profound interpretation of classification algorithm's performance before or alongside their successful application on real (commercial) music.

This paper is organized as follows: In Section 2, we provide an overview of currently available datasets for diverse MIR tasks. In Section 3, we present theoretical and technical backgrounds as well as insights to the implementation details of the algorithmic composition framework, that generated AAM. Section 4 describes the algorithmic composition process. Section 5 shows three example applications for algorithm comparison with our dataset. Ideas for future work are discussed in the concluding Section 6.

2 State-of-the-art audio datasets

Table 1 provides a selection of datasets with ground truth event annotations (without datasets which have only high-level annotations like genres). Please note that we focus only on datasets which have been presented in peer-reviewed publications and contain at least 50 complete music audio tracks.[2]

With regard to the richness of annotations, no individual dataset provides all information available in AAM. Most annotations were created by human experts or mining from the Internet. Some annotations were generated automatically from aligned MIDIs or multitracks (DALI, MedleyDB, MusicNet), some are very precise because of audio generated from MIDIs (MAPS, MSMD, NES-MDB, Slakh). For many datasets, no audio tracks are available, sometimes on request, or as Youtube video clips. Only three datasets contain multitracks (Hargreaves, MedleyDB, Slakh).

Further restrictions include a rather small number of tracks (only five datasets contain more than one thousand tracks), but also genre limitations: GiantSteps contains only electronic dance music, Harmonix only Western pop songs, Isophonics Beatles only Beatle songs, MAPS and MSMD piano pieces, NES-MDB game soundtracks. Even if it is hard to assign genres to artificially generated AAM tracks, the generation process was based on very different music properties (like instruments, tempo, and key, see Section 4). That is why AAM provides more variation than datasets with tracks of only a few genres or artists.

The only dataset which has most similar richness of annotations while also providing (synthesized) audio multitracks is Slakh. In contrast to our dataset, Slakh is built on existing MIDIs manually gathered from the Internet. AAM is based on automatic composition and, therefore, can be easily extended with further tracks adjusting desired properties for specific investigations about when and why algorithms fail. Actually, both datasets can be used in a complementary way, e.g., optimizing a neural network for instrument recognition using both AAM and Slakh tracks.

[1] Find our dataset at: https://doi.org/10.5281/zenodo.5794629

[2] For a more exhaustive (and regularly updated) list, see https://github.com/ismir/mir-datasets (accessed: 02/02/2023).

Table 1 List with selected state-of-the-art music datasets with distinct annotations. The $ symbol marks proprietarily licensed music tracks which have to be purchased

		Annotations									Audio	
Name	**No.**	**Onsets**	**Pitches**	**Instruments**	**Key**	**Tempo**	**Segments**	**Melody**	**Chords**	**Beats**	**Mix**	**Tracks**
AAM	3000	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ACPAS [2]	2189	✓	✓	-	✓	-	-	-	-	-	✓	-
ASAP [3]	520	✓	✓	-	✓	✓	-	-	-	✓	✓	-
CASD [4]	50	-	-	-	-	-	-	-	✓	-	-	-
Chordify TapCorrect [5]	101	-	-	-	-	-	-	-	-	✓	✓	-
DALI [6]	5358	-	-	-	-	-	-	✓	-	-	✓	-
GiantSteps Key [7]	604	-	-	-	✓	-	-	-	-	-	✓	-
GiantSteps Tempo [7]	664	-	-	-	✓	✓	-	-	-	-	✓	-
Hargreaves [8]	104	-	-	-	-	-	✓	-	-	-	✓	✓
Harmonix [9]	912	-	-	-	-	✓	✓	-	-	✓	✓	-
INRIA RWC Pop [10]	100	-	-	-	-	-	✓	-	-	-	$	-
Isophonics Beatles [11]	180	-	-	-	✓	✓	✓	-	✓	✓	-	-
JAAH [12]	113	-	-	-	✓	-	✓	-	✓	✓	-	-
Jamendo VAD [13]	93	-	-	-	-	-	✓	-	-	-	✓	-
MAPS [14]	238	✓	-	-	✓	✓	-	-	✓	-	✓	-
McGill Billboard [15]	740	-	-	-	-	-	-	-	✓	-	-	-
MedleyDB [16]	122	-	-	✓	-	-	-	✓	-	-	✓	✓
MSMD [17]	497	✓	-	-	-	-	-	-	✓	-	-	-
MUSDB18 [18]	150	-	-	✓	-	-	-	-	-	-	✓	✓
MusicNet [19]	330	✓	✓	✓	-	-	-	-	-	-	✓	-
NES-MDB [20]	5000	✓	✓	✓	-	-	-	-	-	-	✓	-
Rock Corpus [21]	200	-	-	-	-	-	-	✓	✓	✓	-	-
SALAMI [22]	1383	-	-	-	-	-	✓	-	-	-	-	-
Seyerlehner Pop [23]	545	-	-	-	-	✓	-	-	-	-	✓	-
Slakh [24]	2100	✓	✓	✓	✓	✓	-	-	-	-	✓	✓
SPAM [25]	50	-	-	-	-	-	✓	-	-	-	-	-

3 AAM dataset

3.1 Motivation and musicological background

As stated in the introduction, artificial music pieces can assist testing and validating classification algorithms with respect to many musically meaningful aspects while easily allowing for automatic and precise annotations. However, such artificial data must stand comparison and compatibility with real (human-composed) music at least to some extent.

When computer supported music classification and analysis is intended to perform high-level tasks, it inherently attempts to emulate human perception. Therefore, the natural human auditory ability of performing grouping mechanisms [26] is involved. For example, the highest notes of a song are usually perceived as one coherent *melody line* (simultaneous grouping), while a melody inherently consists of successive notes (successive grouping). Respectively, multiple notes played at the same time are peceived as one coherent chord and whole music pieces are typically segmented into individual parts that are perceived as entities. Listeners perceive *well-founded Gestalts* ([27], p. 256) which are entities (those very melodies or chords) that become "more than the sum of its parts (notes)" ([28], p. 75). For example: once heard, the Gestalt of a melody will be easily recognized even if its original notes are transposed.

Playing around with these grouping sensations, that listeners do experience intuitively, is the task of a music composer. Therefore, our approach of algorithmic composition keeps in mind those elementary rules of auditory Gestalt-perception ([28], p. 76) by defining specially designed subtasks. We argue that if Gestalts are (easily) perceivable by humans in the resulting artificial music pieces, they should also be identified by pattern recognition algorithms which can be compared and further tuned using our dataset as a testbed.

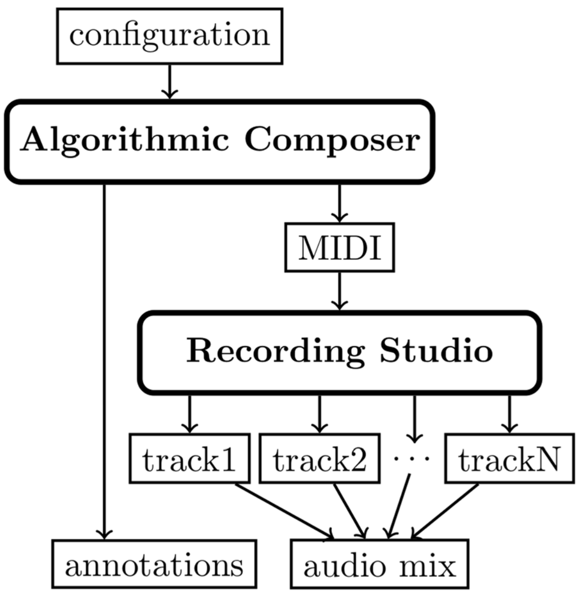

Fig. 1 Modules of the fully automated generation process. The AAM dataset contains all configurations, symbolic MIDI files, single instrument audio tracks, mixed tracks and annotations

3.2 Framework overview

Figure 1 sketches the framework[3]. Beside the final audio mixes and corresponding annotations of onsets, pitches, instruments, keys, tempos, and segments, single tracks of all individual instruments are also included in the dataset in order to enable researchers to perform dataset augmentation by remixing the tracks with different levels of loudness for each instrument and with effects like reverb, chorus, or distortion. Further, we provide all artificially composed MIDI files (e.g., for resynthesis of audio) and the configuration files that state compositional settings.

The *Algorithmic Composer* simulates the composition of one individual music piece. The algorithm constructs relations between musical elements in terms of repetition, variation, and contrast [29], in order to build a consistent piece of music, which fulfills the demands of producing well-founded Gestalts. The task of algorithmic composition is not at all trivial and a topic of many research investigations [30]. Many of them try to compete with great human composers with growing success. Since our main goal is to create data for algorithm evaluation and optimization, our composition approach does not satisfy an ambition to create high-quality music for listening. Nevertheless, the central demand is reducing the portion of unrealistic and poor composed pieces. Furthermore, our work provides a framework designed to grow over time and allowing for successive improvements of the resulting music compositions with future releases.

Beside the *Algorithmic Composer* module, the system's workflow includes the conversion of the music pieces from their symbolic MIDI form to an audio recording, which should be as realistic as possible. This is performed by the module named *Recording Studio*. Each run of the *Algorithmic Composer* composes one music piece. Afterwards, the created piece is converted to single instrument audio tracks using real instrument samples. This single track procedure is similar to music production in a professional recording studio. Finally, all single instrument tracks are mixed down to form a full audio representation (the final mix) of each music piece.

3.3 Technical backgrounds

The whole project is implemented in *Java*. The low-level operations like onset annotation are implemented using *Java Sound API* [31]. To handle musical elements on a higher-level representation than MIDI, the programming library *JFugue* [32], that itself builds on Java Sound, is utilized. It provides the MIDI abstraction language *Staccato*[4], in which all features of the MIDI standard [33] are represented by string tokens. Furthermore, JFugue allows for manipulation of parts of the music as a whole, like sequencing and layering, and therefore simplifies the flexible arrangement of single elements within a full composition. The conversion from the Staccato format into MIDI files with JFugue proved to be reliable and straightforward.

The key aspect in the realistically recording of single instrument tracks is to avoid static sound samples and to use intelligent samplers, which aim to simulate human playing behavior instead. Most samplers used for AAM are part of the *Native Instruments Komplete 11 Ultimate* bundle [34]. Only the guitars were produced with *Chris Hein Guitars* samplers [35], and the non-Western instruments come from the *Ethno World 5* library [36].

For batch conversion (*Recording Studio*), we decided to use a digital audio workstation as used by professional music producers. We chose Harrison's *Mixbus v6* [37], because it combines a high-quality analog mixing simulation with scripting features, which were inherited from its open-source ancestor *Ardour* [38]. With its integrated console for running *Lua* scripts [39], we were able to batch convert our instrument tracks stored as MIDI files into audio files with a CD-standard sampling rate of

[3] The source-code of the composing framework is publicly available at: https://doi.org/10.5281/zenodo.7599941

[4] For easy-to-understand examples of Staccato visit: http://www.jfugue.org/examples.html (accessed: 02/02/2023).

Table 2 List of samplers used to produce the dataset ordered by instrument categories and band functions

Category	Melody	Chords/arpeggio	Bass
Bowed	Erhu, Jinghu, Morin Khuur, Viola, Violin	Cello	Double Bass (pizz/arco)
Brass	Flugelhorn, Trombone, Trumpet	—	—
Flute/Pipe	Concert, Fujara, Pan, Shakuhachi	—	—
Guitars	Electric Guitar (lead)	Acoustic Guitar, Balalaika, Electric Guitar (clean/crunch), Sitar, Ukulele	Electric Bass
Keys	—	Electric Piano, Piano (grand/bright)	Organ
Sax	Alto Sax, Tenor Sax, Clarinet	—	—

44.1 kHz. For compression we used the free lossless audio codec FLAC [40].

4 Algorithmic Composer

4.1 Overview

For practical reasons, constraints are defined in *Algorithmic Composer* to limit the space of possible music pieces and to preserve a manageable complexity. Currently, the *Algorithmic Composer* generates pieces based on rules typical for the composition of Western popular music. For this reason, the music contains only steady rhythms, time signature of 4/4, ordinary major-minor system chord progressions, easily singable melodies, and a moderate number of different instruments per piece. On the other side, the variety of music pieces is extended with several non-Western instruments (cf. Table 2), adding a world music element to Western pop music style compositions. Please note that the limitations in the composition process will be reduced in future releases of the dataset.

For now, the composition process is mainly based on mathematical models relying on random decisions under consideration of knowledge-based rules. Since one music piece is expected to have different parts, which could, for instance, be identified by an automatic segmentation algorithm (cf. Section 5.1), the *Algorithmic Composer* creates at minimum three and at maximum five such parts. The bottom limit of three follows the pop music "rule" that every "good song" is composed of at least chorus, verse, and bridge [41]. This principle is based on the much older compositional concept of *repetition*, *variation*, and *contrast* [29] and can be perceived as Gestalts transforming over time (cf. Section 3.1).

The parameters that define the musical characteristics of the first part of a music piece are selected by chance from a uniform distribution and are bound to the main configuration as listed in Table 3. These parameters essentially determine the general appearance and the degree of variation within and between the resulting music pieces.

Key features of any part are its *key*, *tempo*, *length* (the number of bars), and *instruments*. They are chosen by chance already allowing for a large variety of combinations in the current setting. The probabilities are restricted by the configuration, so that in AAM a key is randomly selected from all twelve root notes in either major or minor tonality. The tempo is drawn as equally-distributed random integer between 60 and 180 BPM. Predominantly, the bar count of parts in popular music is powers of 2. Therefore, we chose 4, 8, and 16. 6 and 12 were added as commonly occurring variations. Other lengths, especially odd-numbered ones, were not allowed for the creation of AAM (cf. Table 3).

4.2 Memoization of random decisions

Because we mainly rely on random decisions, we need a strategy to overcome pure arbitrariness. An example for the limited expression of mere random composition is Mozart's *Musical Dice Game* [42]. The roll of a dice decides for each bar which one of the precomposed phrases is to add to the composition. But, the music shows an unrelated and incoherent organization which appears to be just haphazard especially to inexperienced listeners. This is because a decision to choose a musical phrase does not respect any of the previous decisions.

Table 3 Basic configuration parameters used to produce AAM

Parameter	Values
Duration	120–180 s
Number of parts	3, 4, 5
Part lengths in bars	4, 6, 8, 12, 16
Key roots	All 12
Key tonalities	Major, minor
Tempo range	60–180 BPM
Instruments forget	0.33
Arpeggio probability	0.5
Bass probability	0.9
Chord pad probability	0.8
Drums probability	0.8
Melody probability	0.9

Our approach to this problem is to embrace concepts from the field of *probabilistic programming* [43]. Every random decision that is made, e.g., the choice of key or tempo, is *memoized*, which manifests different *world states* [44]. Memoization originally terms a computational speed optimization technique in *dynamic programming* ([45], p. 365) that relies on caching previous function call calculations for already used input parameters. In probabilistic programming, where function calls with the same parameters can lead to different results, this concept sets probabilities to fixed values, that will from then on not change until the program terminates (in our case: by termination we have generated one piece of music). This procedure is not describable by a Markov model or process, since it does not satisfy the Markov property, which demands that probability distributions are independent from previous states. Instead, it follows a Bayesian approach of conditional probability chains, while preserving an adequate coding workflow [46]. Decision making (in the compositional process) has impact on later decisions, which can lead to (case 1) repeating previous decisions, (case 2) relating to them, or (case 3) deliberately ignoring them. This threesome again meets the former mentioned compositional demands for repetition, variation, and contrast.

One simple example of decision memoization is the generation and variation of the musical key. First, the key for the first part A is randomly drawn with equal probability from all roots and tonalities within the configuration's range, e.g., G minor. Then, the key of the following part B is drawn using adjusted (conditional) probabilities for all possible root notes with relation to the former key from A . These relations are mainly based on the natural harmonics series [47] with a subjectively added preference of using step-up modulation, which is a typical pop music climax technique.[5] Hence, there will be a probability of 0.6 to remain within the key of G minor as established in A (case 1) and different probabilities (case 2) for a modulation to the keys with root notes C (0.1, fourth up), D (0.05, fourth down), B (0.03, major third up), E♭ (0.02, major third down), B♭ (0.03, minor third up), E (0.02, minor third down), A (0.1, whole tone up), and G♯ (0.05, half tone up).

This procedure can be easily transferred to the variation of tempo: With a probability of 0.5, the established tempo is kept in the next part (case 1). With a probability of 0.3, either double or half of the previous tempo is selected (case 2). That agrees with the typical concept of double-time and half-time, which is the major reason for failures of tempo estimators which often produce *tempo octave errors* (integer multiple or fraction of the correct tempo). With a lower probability of 0.1, the tempo is either multiplied by the triplet factor $\frac{2}{3}$ or $\frac{4}{3}$, causing the new tempo to be based on the triplets of the previous beat (also case 2). With the remaining probability of 0.1, the tempo is randomly chosen regardless to the tempo in the previous part (case 3). Besides, a tempo change will never break the configured tempo range constraints.

These design decisions might seem a bit vague and unconfirmed regarding Western pop music, and yes they are. We have not performed formal analysis of real music (yet), so we are unable to concretely prove property distributions in AAM are similar to real music. For instance, the relatively high number of tempo changes in the dataset can be identified as atypical. However, in this case we wanted to provide enough data for the tasks of tempo estimation and detection of boundaries with tempo changes. If one would aim to test a specific algorithm identifying segment boundaries without tempo changes, (s)he can easily identify and omit those specific parts or complete tracks based on the provided annotation. This way, AAM is applicable to many different tasks as we are to show exemplarily in Section 5.

4.3 Element generators

Because the psychology of Gestalt claims that music is perceived by grouped events (cf. Section 3.1), the task of composition is split up in subtasks. We decided to qualify four main roles in a music ensemble: melody, chords, bass, and drums. They are fulfilled by applying independent modules named *element generators* that implement different music composing algorithms. Because different elements are intended to sound together as a coherent part of a music piece, the generators must regard joint properties specified by the part they are currently composing for. These properties are those of key, tempo, and length, which are global for each part. Additionally, an instrument is directly assigned to each element, because the generator must comply to the specific pitch range. Finally, all created elements for one part have to match the same shared chord progression, which is also generated as global part property beforehand.

In the current implementation, chord progressions are constructed as random sequences of chords built from the underlying key's scale degrees of I–VI. The half-diminished chord (VII) is omitted, since it is rarely used in popular music. Furthermore, it would need more sophisticated treatment than the others, so omitting reduces the complexity of the composition process. Each chord is given whole or half note values. The ending chord always becomes the key's tonic. Despite all the simplification, our approach produces solid chord

[5] Parallel upwards key shift is frequently used in pop music compositions, e.g by *Dexys Midnight Runners* when reaching the chorus of "Come On Eileen" or as climax of Michael Jackson's "Man in the Mirror".

movements, which are quite appealing when enriched with a matching melody later on.

The currently implemented element generators are able to produce two kind of chord accompaniment (*chord pads*, *arpeggiations*), simple *bass* lines, reasonable *drum grooves*, and matching *melodies*. The *chord pad generator* arranges the chords from the aforementioned chord progression as triads. It chooses different chord inversions to match a randomly determined range. The *arpeggiation generator* uses the same voicing method. Unlike with the chord pad generator, the resulting notes are not played simultaneously, but in succession with a repeating pattern. The *bass generator* provides the root notes from the chords of the underlying progression. The *drum groove generator* constructs various variations of backbeats. It consists of three instruments: bass drum, snare drum, and cymbal (hihat or ride). The snare and cymbal patterns are randomly selected from a list of precomposed patterns. The bass drum pattern is composed by randomly adding hits to the eighth note positions of a bar (1+2+3+4+) with different probabilities, e.g., $p_{bd}(1) = 1,\ p_{bd}(3+) = 0.3$. For a basic understanding of the functionality of those four generators, we suggest listening to some tracks of the AAM dataset. For details, have a look in the source code. The *melody generator* is the most complex one in the current implementation and will be explained in more detail in Section 4.4.

For the sake of higher variety, the parts of a music piece do not always contain all five elements. Each element gets added to a part with a certain probability (cf. Table 3), so that one part may, for instance, consist of only a melody accompanied by a bass line or just chord arpeggiations with drums. The instrument is, however, *memoized* to its role for all parts of one piece, which means: for all following parts the memoized instrument is likely to be used again, so that, for instance, the main melody instrument probably remains the same for the complete music piece. Since compositional rules are never strict and may be occasionally broken, the memoization is “forgotten” with a probability of 0.33 (“Instruments forget”). Another instrument is then chosen to play the music of that certain element just for that part.

Once all parts for one music piece are produced, they are arranged to form the full structure of the piece. In the current implementation, a fully random sequence is processed, which allows any combination of the parts. The only restriction is that the resulting track must have a playing time between 120 and 180 s. That way, parts are directly repeated multiple times or different parts get inserted in between. That creates *repetition* (AA) and *contrast* (AB). *Variation* (AA’) is implicitly performed by relying on the chain of memoized randomness when composing different parts and elements.

To understand the conceptional construction of the actual music algorithms, the current melody generation procedure is briefly explained in the following section. Please note that other parameter settings are possible, and more generators for any of the four roles can easily be added and work alongside each other due to our framework’s design.

4.4 The melody generator

Melodies are generated by random initialization of eighth notes, and then successively removing, editing, and copying sections until we arrive at a final melody.[6] The classical composition concept of the *period* ([29], p. 55–64) is adopted to guarantee well-formed Gestalts. All decisions are driven by random choices with regard to preceding decisions. Some of the design probabilities are independently drawn each time a construction process starts. These are the melody range and its relation to the last note, which is considered as the melody’s *target note*. The others are memoized until the *Algorithmic Composer* terminates. This way, a coherence throughout all melodies of one music piece is established.

At first, the target note is chosen as the root ($p = 0.75$) or the fifth ($p = 0.25$) of the last chord (target chord) from the underlying progression. The melody is mainly constructed around that note within a randomly chosen range of a fourth, fifth, or sixth. All eight notes of all bars are filled with random notes from inside that range, that diatonically match the given key of the part. In the example melody from Fig. 2:1a, the target note is chosen to be f’ (the root of chord `F`). The melody range is chosen as a sixth ranging from c’ to a’ resulting here in a pool of notes that is {c’,d’,e’,f’,g’,a’}. Afterwards, some of the notes are replaced by rests (Fig. 2:1b). The probability for a rest is a random value p_R between 0.1–0.4 (we denote this hereinafter with $p_R = \text{rand}(0.1, 0.4)$). This probability, in contrast to the target note and melody range, is *memoized* for all following melody generator runs within the composition of one music piece. The idea behind the random choice of probabilities with a range restriction is, that various music pieces can have distinct manifestations of characteristics, while one piece’s melodies remain similar to each other resulting in homogeneous appearance. This is an example for creating a specific *world state* (cf. Section 4.2), in which melodies will (henceforth) always have specifically similar characteristics.

In the second step (Fig. 2:2), the melodic contour is smoothened by removing syncopated notes with a memoized probability $p_s = \text{rand}(0.1, 0.9)$. Third (Fig. 2:3),

[6] The concrete implementation of the melody generator can be found in the source code: `src/parts/MelodyBow.java`

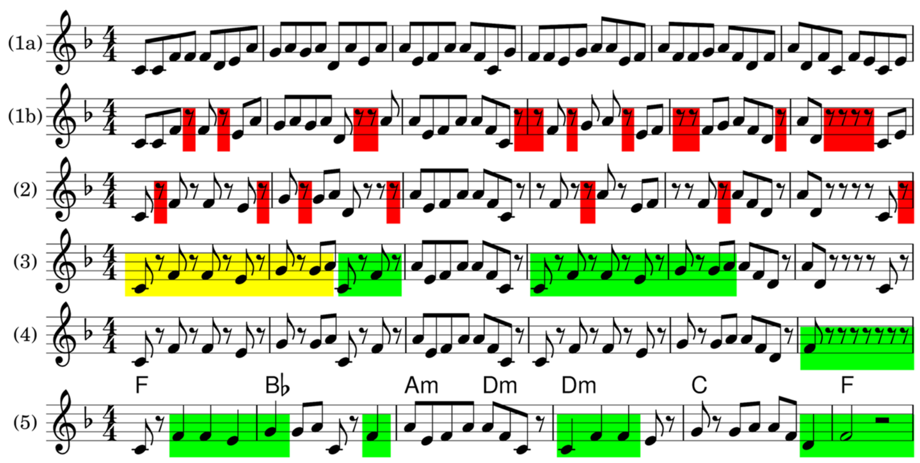

Fig. 2 Step-wise construction of a melody. Red areas mark note removals, yellow areas are notes to copy, and green areas mark replacements

the melody construction is divided into four equal-sized sections. The starting notes from the first section (yellow) are eventually copied over (green) to the second and/or third section by chance. This is done without memoization. The length in eighths of the copy content is randomly chosen for both sections and may become zero (no relation) or even exceed the parts boundary by two eights. This way, resulting forms like ABAC, AABC, AA'A'B, and similar ones are possible. The melody in Fig. 2 could be described as AA'AB.

Fourth (Fig. 2:4), the content of the final bar is replaced by the initially chosen target note. Finally (Fig. 2:5), the melody that up to here contains only eighth notes is altered to quarter and half notes with memoized probabilities of $p_q = \text{rand}(0.1, 0.8)$ and $p_h = p_q + \text{rand}(-0.2, 0.2)$, but only if an extension is possible without overwriting. This last step (eventually) alters the melody's staccato character towards a more legato one.

5 Example applications

In the following sections, we show three possible applications for our dataset. Please note that we do not aim to solve the selected MIR tasks in a perfect way and exhaustively tune the parameters, but just provide examples how different algorithms can be compared. All of them are conducted for audio signals; however, also the comparison of algorithms for symbolic representations is possible using provided MIDI files. For further examples how shallow and deep classifiers can be applied to solve diverse music classification tasks, we refer to [48–50]. Further studies that make use of AAM include segment detection [51], genre recognition [52], and neural architecture search for instrument recognition [53]. These three provide in-depth examples for possible applications.

5.1 Music segmentation

The task of music segmentation is to identify boundaries between parts of music pieces which are perceived as entities. Thus, it is a binary event detection task: each time frame of a piece can be either a boundary or not. Note that another related task, music structure analysis, additionally assigns segment labels and derives relations between musical segments. For further reading on both tasks, we recommend [54, 55].

Two methods have been applied for music segmentation. The first one utilizes the self-similarity matrix (SSM) which compares all time frames of given length in a music track against each other based on a set of features which represent those frames. The alignment of SSM with the Gaussian checkerboard kernel yields a novelty function whose peaks are considered as segment boundaries. For more details, please see [56]. We have applied the implementation based on [57, 58] for the comparison of individual feature groups used to build the SSM, to show their impact on the performance. The feature sets include the Mel frequency cepstral coefficients (MFCCs) extracted from 512 samples using MIRtoolbox [59], the Mel spectrum (2048 samples / Librosa [60]), the semitone spectrum (2048 samples/NNLS Chroma [61]), and the five peaks of fluctuation patterns (229,376 samples / MIRtoolbox). The second method is a convolutional neural network (CNN) after [62]. Table 4 describes the architecture of layers implemented in Keras.

We used 6-fold cross validation in the experiments: The dataset was randomly split into non-overlapping partitions of 500 tracks. Each partition was used once as a test set to validate the performance of both methods. For the CNN, the remaining part of the dataset was used to train the network (with four partitions) and validate its parameters on one partition by early stopping after

Table 4 Architecture blocks for four networks inspired by previous work. Music segmentation: CNN-GrillSchlüter: adaptation after [62]. Instrument recognition: CNN-AlexNet: adaptation after [63]; CNN-Han: adaptation after [64, 65]; CNN-VGG16: adaptation after [66]. Keras layers: C(*i*,*j*): Conv2D(*i*,*j*); P(*m*,*n*): MaxPooling2D(*m*,*n*); d: Dropout(0.25); D: Dropout(0.5); F: Flatten; G: GlobalMaxPooling2D; R_r: Dense(activation='relu'); R_s: Dense(activation='sigmoid'). The numbers of output neurons are provided in square brackets

CNN-GrillSchlüter
C(8,6)[16] p(3,6)
C(6,3)[32]
F D R_s[128] D R_s[1]
CNN-AlexNet
C(11,11)[32] P(1,2) d
C(5,5)[64] P(1,2) d
C(3,3)[128] C(3,3)[128] C(3,3)[128] G d
R_r[1024] D R_s[9]
CNN-Han
C(3,3)[32] C(3,3)[32] P(1,2) d
C(3,3)[64] C(3,3)[64] P(1,2) d
C(3,3)[128] C(3,3)[128] P(2,2) d
C(3,3)[256] C(3,3)[256] G
R_r[1024] D R_s[9]
CNN-VGG16
C(3,3)[32] C(3,3)[32] P(1,2) d
C(3,3)[64] C(3,3)[64] P(1,2) d
C(3,3)[128] C(3,3)[128] C(3,3)[128] P(2,4) d
C(3,3)[256] C(3,3)[256] C(3,3)[256] G d
R_r[1024] D R_s[9]

Table 5 Evaluation measures for music segmentation with respect to different boundaries. F-measure (*F*), precision (*P*), and recall (*R*) are estimated for all boundaries (A) and boundaries coming with instrument (I), key (K), or tempo (T) change. Abbreviations for the feature groups: *MelS*: the Mel spectrum; *SemS*: the semitone spectrum; *Fluct*: fluctuation patterns. Best values are marked with bold font

Meas.	MFCCs	MelS	SemS	Fluct	CNN
F_A	0.646	0.577	0.615	0.470	**0.914**
P_A	0.568	0.473	0.506	0.397	**0.950**
R_A	0.810	0.803	0.850	0.615	**0.899**
F_I	0.600	0.531	0.571	0.418	**0.925**
P_I	0.484	0.401	0.432	0.322	**0.947**
R_I	0.871	0.870	**0.926**	0.644	0.921
F_K	0.520	0.469	0.510	0.374	**0.822**
P_K	0.399	0.338	0.370	0.274	**0.798**
R_K	0.863	0.887	**0.951**	0.666	0.901
F_T	0.387	0.357	0.377	0.299	**0.846**
P_T	0.272	0.237	0.251	0.201	**0.895**
R_T	0.840	0.912	**0.947**	0.718	0.857

three epochs with no loss reduction. Table 5 lists evaluation measures estimated with a 3 s tolerance boundary. F_A, P_A, R_A report F-measure, precision, and recall for all boundaries. Indices *I*, *K*, *T* correspond to measures estimated only for boundaries with instrument, key, and tempo change, respectively.

The results show that the CNN performs at best with respect to *F* and *P* for all boundary types, but the *R*-values (i.e., the share of correctly predicted boundaries) were better for specific boundaries, when novelty-based segmentation with semitone spectrum was applied. However, this method has predicted far more false positives, so that the precision values were less than half of these values for CNN. When we compare different features for novelty-based segmentation, MFCCs seem to perform at best for *F* and *P*.

5.2 Instrument recognition

Instrument recognition is applied here as a multi-label classification task which predicts occurrences of nine instrument groups in classification time frames of a polyphonic audio signal. Other application scenarios can be the prediction of predominant instruments or their relative contribution to the energy of the mixed signal.

We have applied several CNNs based on [64]. Beyond the adjusted implementation from [65], we have tested the slight changes of architecture inspired by AlexNet [63] (larger convolution filters) and VGG16 [66] (more convolution layers). Table 4 lists details of the layers implemented in Keras. The Mel spectrogram extracted with Librosa [60] was used as input. As a traditional method to compare with, we have applied also random forest with a set of timbre-related features available in AMUSE [67]: zero-crossing rate, spectral characteristics, MFCCs, delta MFCCs, etc., all together 157 dimensions, which were stored separately from two time frames, one with the onset event, and another one in the middle between the current and next onset event.

The six partitions of 500 tracks from the music segmentation experiments are used again. For random forest, the trees were trained with all partitions except the former test partition. For CNNs, 4 of 6 partitions were used for training and one partition for validation; the training was continued until no progress of loss was measured during three epochs. Table 6 reports mean test cross-validation balanced classification errors at the individual prediction of eight instrument groups (the average errors for samples with at least one instrument in the group to predict and samples without such instruments).

Table 6 Balanced classification errors for recognition of instrument groups. CNN-AlexNet: adaptation after [63]; CNN-Han: adaptation after [64, 65]; CNN-VGG16: adaptation after [66]. Best values are marked with bold font[a]

Instr. Group	Random Forest	CNN-Han	CNN-AlexNet	CNN-VGG16
Bass	0.192	0.017	0.021	**0.016**
Brass	0.270	0.043	0.094	**0.035**
Drums	0.159	0.036	**0.028**	0.035
Guitar	0.346	0.106	0.169	**0.095**
Organ	0.101	0.002	0.004	**0.001**
Piano	0.388	0.099	0.162	**0.094**
Pipe	0.418	**0.087**	0.145	0.087
Reed	0.389	**0.106**	0.172	0.117
Strings	0.168	0.030	0.051	**0.029**

[a]CNN-Han is better than CNN-VGG16 for pipe at the 6th position after the fix point

We can observe that random forest performs worst. The architecture inspired by VGG16 is the best candidate, followed by Han; however, the differences in absolute error values are small. The architecture inspired by AlexNet performs worse than both other CNN architectures except for drums.

5.3 Onset detection

Onset detection is a binary event recognition task which identifies time frames with new beginning notes. To illustrate the progress of classification performance of algorithms developed over recent decades, we compare more recent approaches from Librosa [60] and MIRtoolbox [59] (`mironsets` method) with their default parameters to the former setup by [68] in Table 7. Since we annotate MIDI note-on events, all methods were applied in attack detection mode. The tolerance boundary is set to 50 ms.

Beyond the best overall performance of the Librosa method, it performs better for slower songs, while MIRtoolbox and Klapuri tend to work better for faster tempos. The Librosa method benefits from the presence of drums. The outdated method from [68] generally performs worst as expected, which by considering its age reveals the quality leap in research.

6 Concluding remarks and outlook

In this paper, we have presented a large dataset of 3000 artificially generated sample-based audio tracks together with original MIDIs, which were created using an algorithmic composition strategy.

Our generation procedure is intended to produce a large number of audio tracks in a short time. Thus, some "blind spots" of the current state of the dataset, such as constant rhythms, the limited number of instrument bodies, or the bias of melody lines towards Western popular music can be addressed in future work. We might closely analyze or even learn real music's property distributions. The proposed framework is designed to be easily extendable with further music composing modules (element generators) and the source code is publicly available. This offers a great potential for advancements in quality and realism of the generated music. New modules may therefore implement concepts of (knowledge-based) grammars, (bio-inspired) transformation of existing music, (evolutionary) optimization approaches, (deep) machine learning or any imaginable hybridization. Another significant step towards realism would be achieved by adding singing voices, e.g., by automatic generation of accompaniment to pre-recorded human vocals or even the artificial creation of realistic voices. Further experiments may also include data augmentation and combination of AAM with data from other datasets to create more robust classification models.

Table 7 Evaluation measures for onset detection on subsets of the AAM dataset. F-measure (*F*), precision (*P*), and recall (*R*) are provided for all three methods on all available dataset tracks (A), as well as on subsets of specific tempo segments (S: slow, M: medium, F: fast) and segments with and without drums (D, $\overline{\text{D}}$). Best values are marked with bold font

Measure	Kla99	Librosa	MIRtoolbox
F_{A}	0.224	**0.787**	0.515
P_{A}	0.228	**0.834**	0.473
R_{A}	0.226	**0.752**	0.582
F_{S}	0.105	**0.821**	0.424
P_{S}	0.095	**0.861**	0.355
R_{S}	0.122	**0.796**	0.552
F_{M}	0.196	**0.793**	0.506
P_{M}	0.192	**0.841**	0.459
R_{M}	0.205	**0.762**	0.577
F_{F}	0.333	**0.742**	0.599
P_{F}	0.369	**0.806**	0.596
R_{F}	0.311	**0.702**	0.611
F_{D}	0.222	**0.804**	0.517
P_{D}	0.223	**0.830**	0.480
R_{D}	0.225	**0.782**	0.575
$F_{\overline{\text{D}}}$	0.228	**0.673**	0.522
$P_{\overline{\text{D}}}$	0.258	**0.851**	0.473
$R_{\overline{\text{D}}}$	0.218	0.603	**0.625**

Abbreviations

AAM	Artificial Audio Multitracks
CNN	Convolutional neural network
MFCC	Mel frequency cepstral coefficient
MIDI	Musical Instrument Digital Interface

MIR	Music information retrieval
SSM	Self-similarity matrix

Acknowledgements
Not applicable.

Authors' contributions
FO: design, implementation, and description of the music framework; execution of onset detection. IV: overview of existing datasets; design, execution, and evaluation of application experiments. ME: musicological background and consultant on composing strategies. The authors read and approved the final manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL.

Availability of data and materials
Find our dataset at: https://doi.org/10.5281/zenodo.5794629. The source code of the composing framework is available at: https://doi.org/10.5281/zenodo.7599941

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 15 November 2022 Accepted: 3 March 2023
Published online: 23 March 2023

References
1. E. Strickland, Andrew ng, ai minimalist: The machine-learning pioneer says small is the new big. IEEE Spectr. **59**(4), 22–50 (2022)
2. L. Liu, V. Morfi, E. Benetos, in *Society for Music Information Retrieval Conference, ISMIR, Late-breaking Demo 2021*, ACPAS: A dataset of aligned classical piano audio and scores for audio-to-score transcription (2021)
3. F. Foscarin, A. McLeod, P. Rigaux, F. Jacquemard, M. Sakai, in *International Society for Music Information Retrieval Conference (ISMIR)*, ASAP: a dataset of aligned scores and performances for piano transcription (2020), pp. 534–541
4. H.V. Koops, W.B. de Haas, J.A. Burgoyne, J. Bransen, A. Kent-Muller, A. Volk, Annotator subjectivity in harmony annotations of popular music. J. New Music. Res. **48**(3), 232–252 (2019)
5. J. Driedger, H. Schreiber, W.B. de Haas, M. Müller, in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR, Delft, The Netherlands, November 4-8*, Towards automatically correcting tapped beat annotations for music recordings (2019), pp. 200–207
6. G. Meseguer-Brocal, A. Cohen-Hadria, G. Peeters, in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR, Paris, France, September 23-27*, ed. by E. Gómez, X. Hu, E. Humphrey, E. Benetos. DALI: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm (2018), pp. 431–437
7. P. Knees, Á. Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, M.L. Goff, in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR, Málaga, Spain, October 26-30*, ed. by M. Müller, F. Wiering. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections (2015), pp. 364–370
8. S. Hargreaves, A. Klapuri, M.B. Sandler, Structural segmentation of multitrack audio. IEEE Trans. Audio Speech Lang. Process. **20**(10), 2637–2647 (2012)
9. O. Nieto, M. McCallum, M.E.P. Davies, A. Robertson, A.M. Stark, E. Egozy, in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR, Delft, The Netherlands, November 4-8*, ed. by A. Flexer, G. Peeters, J. Urbano, A. Volk. The harmonix set: Beats, downbeats, and functional segment annotations of western popular music (2019), pp. 565–572
10. F. Bimbot, E. Deruty, G. Sargent, E. Vincent, in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, ed. by A. Klapuri, C. Leider. Methodology and resources for the structural segmentation of music pieces into autonomous and comparable blocks (University of Miami, Miami 2011), pp. 287–292
11. C. Harte, Towards automatic extraction of harmony information from music signals. Ph.D. thesis (Queen Mary University of London, UK, 2010)
12. Y. Broze, D. Shanahan, Diachronic changes in jazz harmony: A cognitive perspective. Music. Percept. Interdiscip. J. **31**(1), 32–45 (2013)
13. M. Ramona, G. Richard, B. David, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, Vocal detection in music with support vector machines (IEEE, Las Vegas, 2008), pp. 1885–1888
14. V. Emiya, R. Badeau, B. David, Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. IEEE Trans. Audio Speech Lang. Process. **18**(6), 1643–1654 (2010)
15. J.A. Burgoyne, J. Wild, I. Fujinaga, in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, ed. by A. Klapuri, C. Leider. An expert ground truth set for audio chord recognition and music analysis (University of Miami, Florida, 2011), pp. 633–638
16. R.M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, J.P. Bello, in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, ed. by H. Wang, Y. Yang, J.H. Lee. Medleydb: A multitrack dataset for annotation-intensive MIR research (Taipei, Taiwan, 2014), pp. 155–160
17. M. Dorfer, J. Hajič jr., A. Arzt, H. Frostel, G. Widmer, Learning audio–sheet music correspondences for cross-modal retrieval and piece identification. Trans. Int. Soc. Music. Inf. Retr. **1**(1), 22–33 (2018)
18. Z. Rafii, A. Liutkus, F.R. Stöter, S.I. Mimilakis, R. Bittner. The MUSDB18 corpus for music separation (2017). https://doi.org/10.5281/zenodo.1117372
19. J. Thickstun, Z. Harchaoui, S.M. Kakade, in *5th International Conference on Learning Representations, ICLR*, Learning features of music from scratch (OpenReview.net, Toulon, 2017)
20. C. Donahue, H.H. Mao, J. McAuley, in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR, Paris, France, September 23-27*, The nes music database: A multi-instrumental dataset with expressive performance attributes (2018)
21. T.D. Clercq, D. Temperley, A corpus analysisof rock harmony. Pop. Music. **30**(1), 47–70 (2011)
22. J.B.L. Smith, J.A. Burgoyne, I. Fujinaga, D.D. Roure, J.S. Downie, in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, ed. by A. Klapuri, C. Leider. Design and creation of a large-scale database of structural annotations (University of Miami, Florida, 2011), pp. 555–560
23. K. Seyerlehner, G. Widmer, D. Schnitzer, in *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR*, ed. by S. Dixon, D. Bainbridge, R. Typke. From rhythm patterns to perceived tempo (Austrian Computer Society, Vienna, 2007), pp. 519–524
24. E. Manilow, G. Wichern, P. Seetharaman, J. Le Roux, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity (2019)
25. O. Nieto, J.P. Bello, in *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR, New York City, United States, August 7-11*, ed. by M.I. Mandel, J. Devaney, D. Turnbull, G. Tzanetakis. Systematic exploration of computational music structure research (2016), pp. 547–553
26. A.S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound* (Bradford Books, MIT Press, Cambridge, 1990)
27. C. Stumpf, *Erkenntnislehre*, vol. 2 (S. Hirzel Verlag, Leipzig, 1940)
28. M. Ebeling, in *Music Data Analysis: Foundations and Applications*, ed. by C. Weihs, D. Jannach, I. Vatolkin, G. Rudolph. Musical structures and their perception. (Chapman & Hall/CRC Computer Science & Data Analysis Taylor & Francis Group, New York, 2019)
29. C. Kühn, *Formenlehre der Musik* (Bärenreiter Verlag, Kassel, 1998)
30. A. McLean, R.T. Dean, *The Oxford handbook of algorithmic music* (Oxford University Press, New York, 2018)
31. Oracle, *Java Documentation – Java Sound Programmer Guide* (1993/2001). https://docs.oracle.com/javase/8/docs/technotes/guides/sound/programmer_guide/. Accessed 02 Feb 2023
32. D. Koelle, *JFugue* (2002–2020). http://www.jfugue.org. Accessed 2 Feb 2023
33. MMA, *MIDI Manufacturers Association – Musical Instrument Digital Interface* (1983). https://midi.org/specifications. Accessed 02 Feb 2023

34. Native Instruments, *Komplete 11 Ultimate* (Native Instruments North America Inc, Los Angeles, 2016)
35. C. Hein, *Guitars* (Wizardmedia GmbH, Cologne, 2006)
36. M. Barsotti, *Ethno World 5 - professional & VOICES* (Best Service GmbH, Munich, 2010)
37. Harrison Consoles, *Mixbus v6* (Harrison Audio Consoles, 2009–2023). https://harrisonconsoles.com/product/mixbus/. Accessed 02 Feb 2023
38. P. Davis, *Ardour – the digital audio workstation* (2006–2023). https://ardour.org/. Accessed 02 Feb 2023
39. R. Ierusalimschy, W. Celes, L.H. de Figueiredo, *Lua – the programming language* (Pontifical Catholic University, Rio de Janeiro, Brazil, 1993–2023). https://www.lua.org/. Accessed 02 Feb 2023
40. J. Coalson, *Free Lossless Audio Codec (FLAC)* (Xiph.Org Foundation, 2000–2023). https://xiph.org/flac/. Accessed 02 Feb 2023
41. R. von Appen, M. Frei-Hauenschild, AABA, refrain, chorus, bridge, prechorus : song forms and their historical development. Samples (GfPM). **13**(1), 1–83 (2015). http://www.gfpm-samples.de/Samples13/appenfrei.pdf
42. W.A. Mozart, Anleitung zum Componieren von Walzern vermittels zweier Würfel, in *Köchelverzeichnis, KV 294d/516f.* (Breitkopf & Härtel, Wiesbaden, 1862)
43. A. Pfeffer, *Practical Probabilistic Programming* (Manning, New York, 2016)
44. N.D. Goodman, J.B. Tenenbaum, T. Gerstenberg, in *The Conceptual Mind: New Directions in the Study of Concepts*, ed. by E. Margolis, S. Laurence. Concepts in a probabilistic language of thought (MIT Press, Cambridge, 2014)
45. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, in *Introduction to Algorithms*, Advanced design and analysis techniques: Dynamic programming (MIT Press, Cambridge, 2009), chap. 15
46. N. Goodman, V. Mansinghka, D. Roy, K. Bonawitz, J. Tenenbaum, in *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, vol. 2008. Church: A language for generative models (2008), pp. 220–229
47. P. Hindemith, *The Craft of Musical Composition: Book 1—Theoretical Part* (Schott & Co., London, 1937,1942)
48. Z. Fu, G. Lu, K.M. Ting, D. Zhang, A survey of audio-based music classification and annotation. IEEE Trans. Multimedia. **13**(2), 303–319 (2011)
49. S. Hershey, S. Chaudhuri, D.P.W. Ellis, J.F. Gemmeke, A. Jansen, R.C. Moore, M. Plakal, D. Platt, R.A. Saurous, B. Seybold, M. Slaney, R.J. Weiss, K.W. Wilson, in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, CNN architectures for large-scale audio classification (New Orleans, 2017), pp. 131–135
50. J. Pons, O. Nieto, M. Prockup, E.M. Schmidt, A.F. Ehmann, X. Serra, in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, End-to-end learning for music audio tagging at scale (2018), pp. 637–644
51. I. Vatolkin, F. Ostermann, M. Müller, in *Proceedings of the Genetic and Evolutionary Computation Conference*, An evolutionary multi-objective feature selection approach for detecting music segment boundaries of specific types (GECCO, 2021), pp. 1061–1069
52. I. Vatolkin, M. Gotham, N.N. López, F. Ostermann, in *International Conference on Artificial Intelligence in Music, Sound, Art and Design (EvoMUSART)*, Musical genre recognition based on deep descriptors of harmony, instrumentation, and segments (Springer International Publishing, 2023). Accepted
53. L. Fricke, I. Vatolkin, F. Ostermann, in *International Conference on Artificial Intelligence in Music, Sound, Art and Design (EvoMUSART)*, Application of neural architecture search to instrument recognition in polyphonic audio (Springer International Publishing, 2023). Accepted
54. J. Paulus, M. Müller, A. Klapuri, in *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR*, ed. by J.S. Downie, R.C. Veltkamp. State of the art report: Audio-based music structure analysis (2010), pp. 625–636
55. O. Nieto, G.J. Mysore, C. Wang, J.B.L. Smith, J. Schlüter, T. Grill, B. McFee, Audio-based music structure analysis: Current trends, open challenges, and applications. Trans. Int. Soc. Music Inf. Retr. **3**(1), 246–263 (2020)
56. M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications* (Springer, New York, 2015), pp.207–212
57. M. Müller, F. Zalkow, in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR, Delft, The Netherlands*, ed. by A. Flexer, G. Peeters, J. Urbano, A. Volk. FMP notebooks: Educational material for teaching and learning fundamentals of music processing (2019)
58. I. Vatolkin, M. Koch, M. Müller, in Proceedings of the 10th International Conference on Artificial Intelligence, in *Music, Sound, Art and Design (EvoMUSART)*. ed. by J. Romero, T. Martins, N. Rodríguez-Fernández. A multi-objective evolutionary approach to identify relevant audio features for music segmentation (Springer International Publishing, Virtual Event, 2021), pp. 327–343
59. O. Lartillot, P. Toiviainen, in *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR*, ed. by S. Dixon, D. Bainbridge, R. Typke. MIR in Matlab (II): A toolbox for musical feature extraction from audio (Austrian Computer Society, Vienna, Austria, 2007), pp. 127–130
60. B. McFee, C. Raffel, D. Liang, D.P. Ellis, M. McVicar, E. Battenberg, O. Nieto, *in Proceedings the Python Science Conference* (Audio and music signal analysis in python, Librosa, 2015), pp. 18–25
61. M. Mauch, S. Dixon, in *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR*, ed. by J.S. Downie, R.C. Veltkamp. Approximate note transcription for the improved identification of difficult chords (Utrecht, Netherlands, 2010), pp. 135–140
62. T. Grill, J. Schlüter, in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, Music boundary detection using neural networks on combined features and two-level annotations (2015), pp. 531–537
63. A. Krizhevsky, I. Sutskever, G.E. Hinton, in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, ed. by P.L. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger. Imagenet classification with deep convolutional neural networks (2012), pp. 1106–1114
64. Y. Han, J. Kim, K. Lee, Deep convolutional neural networks for predominant instrument recognition in polyphonic music. IEEE ACM Trans. Audio Speech Lang. Process. **25**(1), 208–221 (2017)
65. I. Vatolkin, B. Adrian, J. Kuzmic, in Proceedings of the 10th International Conference on Artificial Intelligence, in *Music, Sound, Art and Design*. ed. by J. Romero, T. Martins, N. Rodríguez-Fernández, A fusion of deep and shallow learning to predict genres based on instrument and timbre features, (Springer, Virtual Event, 2021), pp. 313–326
66. K. Simonyan, A. Zisserman, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, ed. by Y. Bengio, Y. LeCun. Very deep convolutional networks for large-scale image recognition (2015)
67. I. Vatolkin, P. Ginsel, G. Rudolph, in *Proc. 44th Int'l ACM Conf. on Research and Development in Information Retrieval (SIGIR)*, Advancements in the music information retrieval framework AMUSE over the last decade (2021), pp. 2383–2389
68. A.P. Klapuri, in *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 6. Sound onset detection by applying psychoacoustic knowledge (1999), pp. 3089–3092

Publisher's Note