# MIRDATA: SOFTWARE FOR REPRODUCIBLE USAGE OF DATASETS

**Rachel M. Bittner**[*1], **Magdalena Fuentes**[*2,3], **David Rubinstein**[1], **Andreas Jansson**[1]
**Keunwoo Choi**[1], **Thor Kell**[1]
[1] Spotify, USA [2] L2S, CNRS–Univ.Paris-Sud–CentraleSupélec, France
[3] LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[*]Equal contribution

## ABSTRACT

There are a number of efforts in the MIR community towards increased reproducibility, such as creating more open datasets, publishing code, and the use of common software libraries, e.g. for evaluation. However, when it comes to datasets, there is usually little guarantee that researchers are using the exact same data in the same way, which among other issues, makes comparisons of different methods on the "same" datasets problematic. In this paper, we first show how (often unknown) differences in datasets can lead to significantly different experimental results. We propose a solution to these problems in the form of an open source library, `mirdata`, which handles datasets in their current distribution modes, but controls for possible variability. In particular, it contains tools which: (1) validate if the user's data (e.g. audio, annotations) is consistent with a canonical version of the dataset; (2) load annotations in a consistent manner; (3) download or give instructions for obtaining data; and (4) make it easy to perform track metadata-specific analysis.

## 1. INTRODUCTION

Music Information Retrieval (MIR) systems are often software or algorithms which are evaluated and compared based on their performance according to appropriate metrics on chosen datasets. These systems are becoming increasingly complex; reproducing systems presented in academic publications requires access to the software and data [23]. As outlined in [23], some of the common elements of an MIR system are (1) Data (Audio and Annotations) (2) Codecs and Parsing (3) Modeling and (4) Evaluation. The reproducibility of each of these elements poses challenges, but efforts are being made to reduce potential inconsistencies.

For evaluation, different implementations of evaluation metrics can result in substantially different results, motivating the need for `mir_eval` - a common and transparent evaluation software [29]. For modeling, slightly different implementations of the same algorithm can result in very different results [23]. Recently, this has been mitigated by the availability of software with tools for popular MIR tasks. Some examples are `librosa` [25] and `essentia` [6] - tools for MIR related signal processing, simple models and commonly used algorithms; Scikit-Learn [28] - tools for training simple machine learning algorithms; and `madmom` [5] - deep learning and machine learning models for common MIR tasks such as chord recognition, beat and downbeat tracking.

It is very difficult to get licenses to distribute music recordings openly. As a result, the majority of datasets available do not have freely available audio files; the exchange of this data is often done manually, which can result in varying data versions. When working with pairs of audio and annotation files, it is important that the audio files used are the same files that were used to create the annotations. When audio files are released separately from annotations, unknown differences between the original and other versions of the audio can create reproducibility issues. Websites such as Zenodo [1] and Figshare [2] provide permanent hosting and versioning of datasets, increasing reproducibility, but many datasets used in MIR are not available on such websites, and (often unknown) differences in data can adversely affect downstream performance.

Additionally, the annotations that come with each of these datasets exist in a huge variety of formats. Among these formats, some provide very complete information e.g. in the form of a JAMS file [17, 22], while others lack crucial information needed to accurately use the data, such as the time stamps associated with different observations. Most of the time, researchers write their own code for parsing the specific annotation files they use for a particular dataset. This is both inefficient and error-prone; what was found for evaluation and modeling is also true for data parsing: small differences in annotation loading code can result in huge differences in results downstream. Finally, the pairing of audio and annotation files is often done manually each time, usually by matching on filename substrings. In addition to being cumbersome, this can also lead to mismatched audio and annotation files.

To summarize, obtaining datasets and writing code to



[1] `https://zenodo.org`
[2] `https://figshare.org`

process them is both time consuming and error-prone. As a result, researchers are less inclined to use multiple datasets for their task, and instead develop or test their models on single datasets, reducing the reliability of their results [33]. We believe that the two current biggest blockers of reproducibility in the MIR community are (1) the lack of open datasets, resulting in a lack of transparency as to the consistency of data across publications and (2) the lack of an open library for consistently loading annotations in various formats in common datasets.

In this paper, we introduce an open source library, `mirdata`, which provides tools for using common MIR datasets. We aim to create a useful tool for researchers, which will increase reproducibility, and facilitate and encourage the use of several datasets for evaluation. In particular, it contains tools for loading dataset-specific annotations in a consistent manner, validating if a dataset copy the user has is consistent with a canonical version of the dataset, downloading datasets, and linking audio and annotation files along with track level metadata. We demonstrate the need for this tool by highlighting inconsistencies in common practices when loading annotations and in the data itself (annotations and audio) for three popular MIR datasets, namely iKala, Salami, and the Beatles dataset.

The library is publicly available on Github at `github.com/mir-dataset-loaders/mirdata`.

## 2. RELATED WORK

Data utility libraries exist for other fields such as text, video and image analysis [1, 10, 27, 28] which allow a user to download a dataset and load it into memory for ease of use in experimentation and consistent results. TensorFlow [1], a deep learning framework, includes a variety of datasets covering images, text, language translation and video [3]. In order to ensure the integrity of the data, TensorFlow hard codes expected file sizes and SHA256 checksums of each file in their library, as well as paths of the data included with the dataset. If the expected values do not match what is downloaded, the local dataset is not considered valid and is not available for usage in the library.

Scikit-Learn [28], a popular machine learning library for Python, includes their own set of dataset loading utilities [4]. Some small datasets, known as "toy datasets", are included directly in the library. Larger datasets, known as "real-world datasets" are downloaded and stored in a "data home" directory on the local machine. Like TensorFlow, Scikit-Learn checks for dataset integrity based on a SHA256 checksum, but only checks the downloaded zip or tar file itself. This approach requires that users download the entirety of the Scikit-Learn library in order to use the dataset loaders.

MLDatasets [5] acts as a specification for how datasets should be managed. DataDeps [36] specifies dataset metadata that conforms to a management system. This method allows for multiple people to maintain their own dataset repositories or for a single organization to set up multiple, independent libraries, each for one dataset.

There are few examples of dataset utility libraries for music. However, the python library Nussl [19] for source separation contains some dataset utilities. Unlike the other libraries previously mentioned, `Nussl` does not include any utilities for dataset retrieval and expects the user to have the datasets locally on the machine before use. However, it includes expected checksums for the dataset audio and logic to check for validity and existence of the dataset as well as simple utilities for loading the data.

Software also exists for loading particular types of (music-centric) annotations, including `pretti_midi` [6] for MIDI data, JAMS [17] for data released in JAMS format, and Music21 [7] for MIDI and MusicXML data. When annotations are released in these formats, custom loading code is less necessary. However, many annotations are released in other formats and require custom loading code.

## 3. AUDIO FILES IN MIR DATASETS

Datasets in MIR suffer from a unique constraint: most music is protected under copyright. Datasets which are built on copyrighted materials are not typically available for open download. There are several common levels of access for the audio files for different MIR datasets:

1. Open Access
2. Restricted Access (e.g. password protected)
3. "Do it yourself" Access (e.g. YouTube links)
4. No Access

We surveyed 128 MIR datasets from the "Audio Content Analysis" website [8] in April 2019 and determined their access levels. By our estimate, 80 were "open access", 19 were "restricted access", 15 were "DIY" Access, 14 were "no access", meaning that 22.8% of the total list is not openly available. These limited access datasets include historically popular datasets such as RWC [13], AudioSet [12], CAL10k [32], the Beatles dataset [16], iKala [8], the Million Song Dataset [2], and Salami [31].

The more restrictive the access level, the more room there is for "dataset telephone"; when it is difficult to access a particular dataset from a common repository, researchers may share their personal copies with each other, which may contain perturbations from when they first received it. Additionally, since the audio and annotations are sometimes released separately, if the audio is incorrect, the annotations will not correspond to the audio files, resulting in inconsistencies during model development and evaluation. As a result, researchers are performing experiments and computing metrics on datasets they believe are the same as others versions, but may be quite different in reality.

[3] `www.tensorflow.org/datasets/datasets`
[4] `https://scikit-learn.org/stable/datasets`
[5] `https://github.com/JuliaML/MLDatasets.jl`
[6] `github.com/craffel/pretty_midi`
[7] `http://web.mit.edu/music21/doc/index.html`
[8] `https://www.audiocontentanalysis.org/data-sets/`

In an ideal case, the audio files used for a dataset should be the same as those used to create the annotation files. There are a number of popular datasets for which the audio is difficult to obtain. For example, the 7-digital preview clips of the million song dataset [2] have often been used for music classification tasks. While the clips were previously available through an API, it has since been shut down and the clips are no longer available. In the Beatles dataset [16], audio is not released, but instead, catalog numbers and release years of the albums used are provided to prevent differences in audio versions used. Regardless, we found that different versions of the dataset have been used by researchers (see Section 4.1). In the case of AudioSet [12], audio is provided in the form of YouTube video identifiers, adding a new challenge in data reproducibility. However, the availability of the linked YouTube videos changes over time, and accessibility varies by country.

## 4. EXPERIMENTS - WHY A COMMON TOOL IS NEEDED

Differences in audio or annotations, or in the code used to load data into memory can have a huge impact on downstream results. In this section, we examine the effect of real differences we found on evaluation metrics in instances of three popular MIR datasets.

### 4.1 The Beatles Dataset

The Beatles dataset [16] contains annotations for beats, downbeats, sections, and chords for the nearly entire Beatles' collection. However, as the audio is copyrighted, only the annotations are released as part of the dataset. The researchers are asked to use their personal copy of the Beatles' catalog and match the audio files with the annotations.

The annotations were created using a particular version of the audio, and they may not correspond well with other versions. For this dataset in particular, it is quite easy to end up with different versions of the same Beatles track, since there are several releases of every album, including remastered versions.

To evaluate these potential differences, we first compared checksums across four different researcher's copies of the audio files corresponding to the Beatles dataset. Out of the four versions, three had identical checksums, while one had invalid checksums on every single audio file, indicating that the audio is completely different between the two versions. Upon further examination of the differences, we found inconsistencies in the number of channels, the duration, and the average RMS of the audio files between the two versions.

The differences go even further than channels, duration and volume. In Figure 1, we first normalize a pair of audio files to have the same peak level and compute the absolute difference in their spectrograms. In the low frequencies, in particular, there are major differences between the frequency content of the two versions, despite sounding similar.

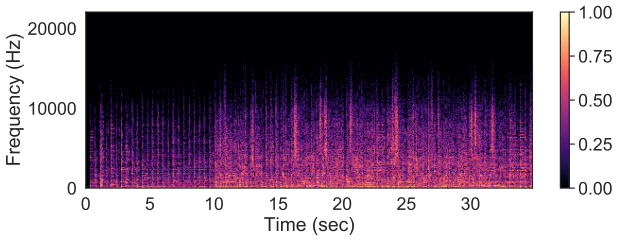


**Figure 1**. Normalized absolute difference between two spectrograms of the first 30 seconds of "Across the Universe" computed on audio files from two versions of the Beatles dataset audio.

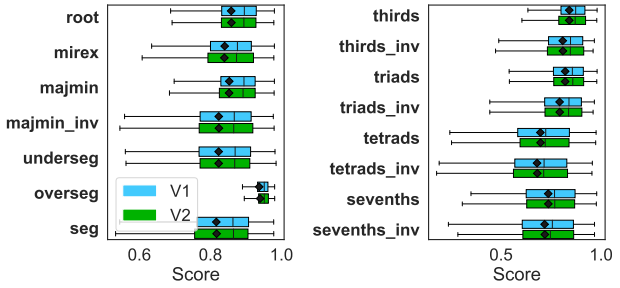


**Figure 2**. Chord metrics for chord estimates computed on two different versions of the Beatles audio, compared with the Beatles dataset reference chord annotations.

Next, we ran a chord recognition algorithm [21] on the two different versions of the audio collection, and computed the standard chord recognition metrics as implemented in `mir_eval` using the dataset's (public) reference chord annotations. The differences in the metrics are shown in Figure 2. While only one of these metrics had statistically significantly different results ("overseg", according to a paired t-test), we see that the same chord recognition algorithm produces results which are different enough to affect the metrics.

### 4.2 The iKala Dataset

The iKala dataset [8] is commonly used for melody estimation, vocal activity detection, and source separation. It contains isolated vocals and instruments (provided as left and right channels of a stereo audio file), along with vocal $f_0$ annotations and lyrics.

We performed the same checksum experiment as for the Beatles dataset and compared checksums for four different researcher's copies of the iKala dataset. We found that all four versions (audio and annotations) were identical.

One challenge with the iKala dataset is that the vocal $f_0$ annotations are provided as newline-separated files with the pitch, but without timestamps, which must be inferred upon load. On the dataset's website, they state that the hop size is 0.032 seconds, but it does not state the alignment of the first time frame (left aligned or center aligned). The dataset's website also provides code for loading the annotation files, which uses a different hop size of 0.03125 seconds and center aligned frames (with the first time stamp

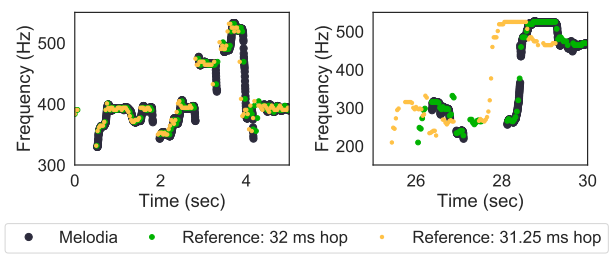


**Figure 3**. iKala reference annotations loaded using two different hop sizes (32 ms and 31.25 ms) versus the output of Melodia. (Left) the first 5 seconds of the track. (Right) the last 5 seconds of the track.

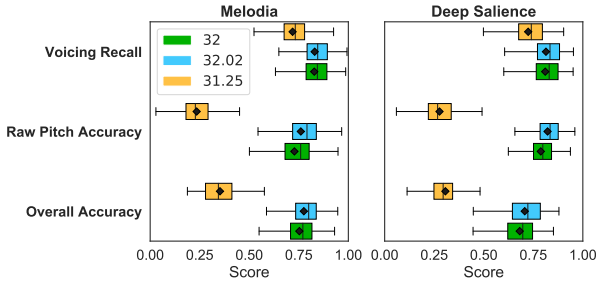


**Figure 4**. Melodia and Deep Salience melody metrics when evaluated against iKala's reference data loaded with 3 different hop sizes.

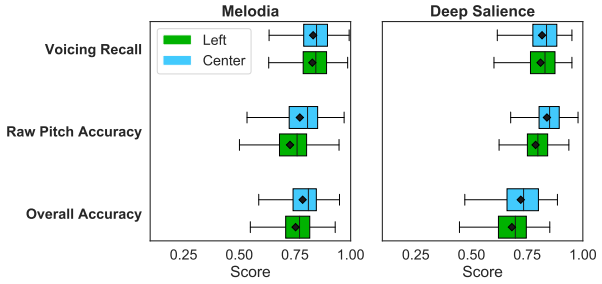


**Figure 5**. Melodia and Deep Salience melody metrics when evaluated against iKala's reference data loaded with left and center-aligned time stamps.

at 0.5 * hop seconds).

To see how users infer the iKala time stamps, we performed a search of public code on Github.com for code which loads the iKala pitch annotations. We found 5 unique ways of loading the time stamps, consisting of 3 different hop sizes (0.032 s and 0.03125 as listed on the dataset website, and 0.032017, inferred from the duration of the audio files), and two different alignments (left and center). By far, the most common combination was using a hop size of 0.032 s and left aligned frames.

The differences in hop size have a major effect on the alignment of the audio and the annotation, especially over time. Figure 3 shows an example of the annotations loaded with two of the hop sizes, and the estimate of a melody extraction algorithm (Melodia [30]) for comparison. In the first 5 seconds, the differences are small, but in the last 5 seconds, we see a visible misalignment between the loaded annotations and the audio.

To investigate the severity of these differences, we ran two melody extraction algorithms, Melodia [30] and Deep Salience [3], on the iKala audio. We then compute melody evaluation metrics using `mir_eval` with reference times computed using the three different hop sizes, $h = 32$ ms, $h = 32.017$ ms and $h = 31.25$ ms, using *left* aligned frames. In Figure 4, we show boxplots of the results across tracks in the dataset for each of these reference hop sizes. The results for different hop sizes are quite different, and drastically so for the smallest hop size, 0.03125 s. Even the difference between $h = 32$ and $h = 32.017$ in Overall Accuracy is substantial for both datasets - a difference that is historically enough to claim state of the art over another algorithm. A paired T-test shows statistically significant differences for all pairs of hop sizes for each metric, with the exception of Voicing Recall for $h = 32$ and $h = 32.017$

Next, we compute the melody metrics for the same melody extraction algorithms using a hop size of $h = 32$ ms and compare left vs center aligned frames in the reference. Figure 5 shows the results per track of the two different reference alignments. The difference in metrics is smaller than for the hop size differences, but left alignment is statistically significantly worse than right alignment for Overall Accuracy and Raw Pitch Accuracy under a paired T-test.

This begs the question: which is the correct way to load the timestamps? Since it is quite unlikely that an incorrectly time aligned reference would produce higher scores than a correct alignment, it is likely that, despite the norm of using left-aligned frames, the annotations are intended to have center aligned timestamps. Indeed, if we look at a specific example of left vs. center aligned timestamps for a short excerpt compared with two different algorithm estimates, as in Figure 6, we see that the reference is better aligned with both estimates when using center aligned time frames. Note that in Figure 4, the reference hop of 32.02 ms resulted in higher metrics than the hop of 32 ms (both with left aligned frames), and a 32 ms hop with center aligned frames has higher metrics than all of the left-aligned hops. The data loaded with a hop size of 32.02 ms starts off misaligned but over time, approaches a center alignment, explaining the "better" score with this incorrect hop size.

The shocking result of this set of experiments is that every single example we found publicly – including the code found on the dataset's website – appears to be loading the $f_0$ data incorrectly, either with an incorrect hop size or an incorrect alignment - no example we found had both a hop size of 32 ms and center alignment.

### 4.3 Salami Dataset

The Salami dataset [31] is a popular dataset used for music structural segmentation. It consists of 1359 tracks across

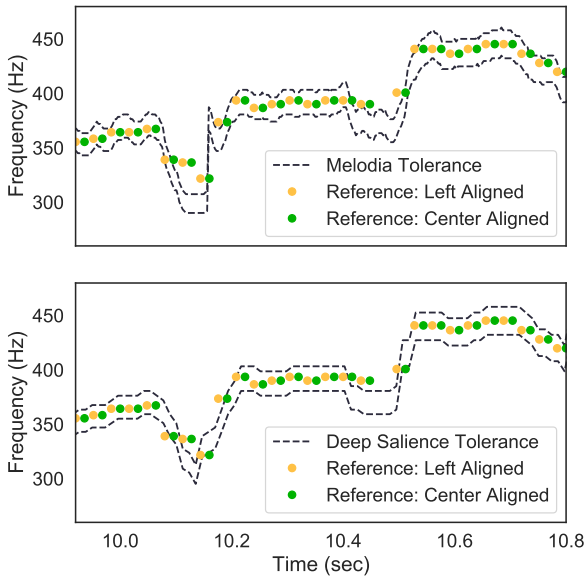


**Figure 6**. Left and center aligned time stamps for a track in iKala versus algorithm estimates from Melodia and Deep Salience. The dashed lines show the distance from the estimate where the algorithm would be considered correct in the standard melody extraction metrics.

a wide variety of genres, namely classical, jazz, popular, world, among others. Each track has annotations of 'coarse' and 'fine' segments, and among the annotated files, a subset of 884 tracks was annotated by two distinct annotators. The complete set of annotations was released in version 2.0 of the dataset, increasing the volume of the dataset with respect to previous versions. For instance, from version 1.9 to 2.0 additional annotations related to 539 tracks were added, 390 with multiple annotations and 189 with single annotations. Both versions are available in the dataset's main repository. However, as in other cases when a dataset is updated, there is no centralized version control that is transparent and ensures the awareness of the community to these changes.

Data-driven models are increasingly popular for addressing MIR tasks, including the task of boundary detection using the Salami dataset [15, 35]. One of the main reproducibility-related issues about data-driven models is their training, in particular, the amount of annotated data is crucial. When using Salami, it is important to avoid using an old version of the annotations, which could have a negative impact in a model's performance in comparison with the same model trained on the newest version of the dataset. In particular, if different data is used for training two different models with the aim of comparing their performance afterward, it is difficult to isolate possible causes of performance differences. An example of this situation is shown in [11], where the authors use a different subset of Salami than previous works, obtaining substantially different results when intending to re-implement other authors' model (e.g. 0.246 instead of the previously reported 0.523 F-measure with a $\pm 0.5$ s tolerance window).

Another possible source of inconsistency with the use of this dataset relates to contributions from people other than the dataset creators. The authors in [24] manually edited 171 of the annotations in version 2.0, to correct formatting errors and enforce consistency with the annotation guide proposed by the dataset creators. However, this "corrected" version of the annotations was not included in the dataset's main repository. This third version of annotations is used in recent works [24, 34]; comparison of systems without awareness of the difference with the version 2.0 released annotations may lead to differences in performance that are beyond models' design.

## 5. MIR DATASET LOADERS

In this section, we describe the `mirdata` python library, our proposed solution to the current reproducibility issues with dataset versions and loaders. A driving philosophy of this library is to work with the imperfect situation we are faced with, with regards to the limited openness of MIR data. In an ideal scenario, all data would be freely sharable and version controlled; since this is not the case, we do our best to create tools to maximize reproducibility given the current constraints. Most importantly, we aimed to create a clean, transparent and easy to use interface to encourage reuse and contributions. The first release of the library will include loaders for Orchset [7], iKala [8], MedleyDB Melody and Pitch subsets [4], the Beatles dataset [16], Salami [31], the Million Song Dataset [2], Medley Solos-DB [18], RWC [9, 13, 14, 20], DALI [26], and GuitarSet [37].

### 5.1 Dataset Indexes and Checksums

Datasets, by their definition, are collections of data. In the case of MIR datasets, the data is often a collection of separate files, some of which correspond to e.g. a particular audio file. For example, the Beatles dataset contains four separate text files containing chord, beat, section, and key annotations for each audio file. Since each of these four annotation files are related to the same audio file, it is desirable to have a common way of linking them. In `mirdata`, we use a *dataset index* to link related files, in the form of a JSON file. This index contains a unique identifier for each group (for example, the name of the audio file), which is mapped to its corresponding file paths and their expected *checksums*, for example:

```
{
    "track1": {
        "audio": [
            "example/audio/track1.wav",
            "912ec803b2ce49e4a541068d495ab570"
        ],
        "annotation": [
            "example/annotations/track1.csv",
            "2cf33591c3b28b382668952e236cccd5"
        ]
    },
    ...
}
```

The use of an index for each dataset is advantageous for a number of reasons. First, it groups related data files in

a transparent way, avoiding audio-annotation pairing mistakes, and removing the need for custom filename substring matching per dataset. Second, it gives a version-controlled record of all expected files in the dataset, preventing inconsistencies due to missing or extra files; we can check if all the expected files are present in a local copy of a dataset, and we load data to memory based on the index, ignoring files not included in it. Finally, it provides a way to verify if a local copy of a dataset is consistent with a canonical version on a file-by-file basis.

For each dataset, we also provide a `validate()` function, which checks for the existence of files locally and compares the expected checksums with checksums of the local copy. A checksum is a representation of a digital file similar to a fingerprint and usually computed by taking a hash of the bits in a file. The smaller (in size) representation created allows for efficient file comparisons at the bit level. If two files have the same checksum then a user can assume that the two files are exactly the same with high confidence. If the checksums between two files differ or the computed checksum is different from an expected checksum, then the user is at a minimum aware of discrepancies and can take appropriate action. In the `mirdata` use case, checksums allow users who have a local copy of a dataset to know whether or not they are using the same data as others, simply by running the validation function.

Note that there is not always one "correct" version of a dataset, so it is difficult to decide which version of a dataset should be used to create the reference checksums. For this library, we compute checksums on the version that is as close to the "original" as possible, for example by obtaining a version from dataset creators.

### 5.2 Dataset Downloading

It can often be difficult or unclear how to get access to a particular dataset, and same data often exists in multiple places and may not be identical. In `mirdata`, we provide a `download()` function for each dataset. When data is openly available online, the function automatically downloads the data, and this version of the data is same the version used to create the checksums. When the data has been downloaded, we run validation to ensure it matches and warn the user if not (e.g. in the case of an incomplete download). When data is not openly available online, we provide instructions for how to obtain the data (e.g. by requesting access on a particular website). Once the data has been obtained, the user can then run validation to ensure the data is consistent.

### 5.3 Annotation Loaders

As highlighted previously, differences in implementations for loading annotation data to memory can have big effects on the resulting data. In `mirdata`, we remove the need for users to manually write loaders per dataset and annotation type by providing functions for loading all annotations for each dataset. These implementations are shared and transparent, allowing users to permanently correct mistakes in the way data is loaded.

As an example, for some of the beat annotations in the Salami dataset, the beat position is missing for only a few observations. These missing positions can be inferred from the neighboring information (e.g. beats 1 and 3 have labels, and the one in between is absent), and in `mirdata`'s implementation we fill in this information on load.

### 5.4 Track Metadata

More often than not, in addition to data files containing e.g. time varying annotations, datasets provide track-level metadata. When loading annotations, we also link any available track level metadata with each track id group. This can be particularly useful when splitting data, such as for creating unbiased train-validation-test splits [23], or for analyzing evaluation metrics over different splits of a dataset.

## 6. CONCLUSIONS

Although data distribution challenges remain, we believe that the use of `mirdata` will result in reproducible usage of datasets in research moving forward. Future iterations of `mirdata` will include support for large (out of memory) datasets and an increased number of supported datasets. As datasets become more open and annotation formats standardize, the scientific need for this library will lessen, but it will remain a useful tool for ease of working with datasets.

Importantly, we designed `mirdata` to have a low barrier to entry for contributions. New datasets can be easily included independently with minimal interfacing with the rest of the library. With active community participation, we believe that `mirdata` can help ensure that MIR datasets are used in a consistent, reproducible manner moving forward.

## 7. REFERENCES

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*, pages 591–596, 2011.

[3] Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan P. Bello. Deep salience representations

for f0 estimation in polyphonic music. In *18th International Society of Music Information Retrieval (ISMIR) Conference*, October 2017.

[4] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan P Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *International Society of Music Information Retrieval (ISMIR)*, October 2014.

[5] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new python audio and music signal processing library. In *Proceedings of the 24th ACM International Conference on Multimedia*, ACMMM, 2016.

[6] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Perfecto Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *14th Conference of the International Society for Music Information Retrieval (ISMIR)*. International Society for Music Information Retrieval (ISMIR), 2013.

[7] Juan J Bosch, Ricard Marxer, and Emilia G'omez. Evaluation and combination of pitch estimation methods for melody extraction in symphonic classical music. *Journal of New Music Research*, 45(2):101–117, 2016.

[8] Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Roger Jang. Vocal activity informed singing voice separation with the ikala dataset. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 718–722. IEEE, 2015.

[9] Taemin Cho and Juan P Bello. A feature smoothing method for chord recognition using recurrence plots. In *12th International Society for Music Information Retrieval Conference*, ISMIR, 2011.

[10] François Chollet et al. Keras. `https://keras.io`, 2015.

[11] Alice Cohen-Hadria and Geoffroy Peeters. Music structure boundaries estimation using multiple self-similarity matrices as input depth of convolutional neural networks. In *Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. Audio Engineering Society, 2017.

[12] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[13] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Popular, classical and jazz music databases. In *ISMIR*, volume 2, pages 287–288, 2002.

[14] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Music genre database and musical instrument sound database. 2003.

[15] Thomas Grill and Jan Schlüter. Music boundary detection using neural networks on combined features and two-level annotations. In *ISMIR*, pages 531–537, 2015.

[16] Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, 2010.

[17] Eric J Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M Bittner, and Juan Pablo Bello. Jams: A json annotated music specification for reproducible mir research. In *ISMIR*, pages 591–596, 2014.

[18] Vincent Lostanlen, Carmine-Emanuele Cella, Rachel Bittner, and Slim Essid. Medley-solos-DB: a cross-collection dataset for musical instrument recognition, September 2018.

[19] Ethan Manilow, Prem Seetharaman, and Bryan Pardo. The northwestern university source separation library. Proceedings of the 19th International Society of Music Information Retrieval Conference (ISMIR 2018), Paris, France, September 23-27, 2018.

[20] Matthias Mauch, Hiromasa Fujihara, Kazuyoshi Yoshii, and Masataka Goto. Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *ISMIR*, ISMIR, 2011.

[21] Brian McFee and Juan P. Bello. Structured training for large-vocabulary chord recognition. In *18th International Society for Music Information Retrieval Conference*, ISMIR, 2017.

[22] Brian McFee, Eric J Humphrey, Oriol Nieto, Justin Salamon, Rachel Bittner, Jon Forsyth, and Juan P Bello. Pump up the jams: V0. 2 and beyond. *Music and Audio Research Laboratory, New York University, Tech. Rep*, 2015.

[23] Brian McFee, Jong Wook Kim, Mark Cartwright, Justin Salamon, Rachel M Bittner, and Juan Pablo Bello. Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research. *IEEE Signal Processing Magazine*, 36(1):128–137, 2019.

[24] Brian McFee, Oriol Nieto, Morwaread M Farbood, and Juan Pablo Bello. Evaluating hierarchical structure in music annotations. *Frontiers in psychology*, 8:1337, 2017.

[25] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[26] Gabriel Meseguer-Brocal, Alice Cohen-Hadria, and Geoffroy Peeters. Dali: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm. In *19th International Society for Music Information Retrieval Conference*, 2018.

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[29] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. mir_eval: A transparent implementation of common mir metrics. In *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.

[30] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio. Speech. Lang. Processing*, 20(6):1759–1770, 2012.

[31] Jordan Bennett Louis Smith, John Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie. Design and creation of a large-scale database of structural annotations. In *ISMIR*, volume 11, pages 555–560. Miami, FL, 2011.

[32] Derek Tingle, Youngmoo E Kim, and Douglas Turnbull. Exploring automatic music annotation with acoustically-objective tags. In *Proceedings of the international conference on Multimedia information retrieval*, pages 55–62. ACM, 2010.

[33] Antonio Torralba, Alexei A Efros, et al. Unbiased look at dataset bias. In *CVPR*, volume 1, page 7. Citeseer, 2011.

[34] Christopher J Tralie and Brian McFee. Enhanced hierarchical music structure annotations via feature level similarity fusion. *arXiv preprint arXiv:1902.01023*, 2019.

[35] Karen Ullrich, Jan Schlüter, and Thomas Grill. Boundary detection in music structure analysis using convolutional neural networks. In *ISMIR*, pages 417–422, 2014.

[36] Lyndon White, Roberto Togneri, Wei Liu, and Mohammed Bennamoun. Datadeps.jl: Repeatable data setup for replicable data science. *CoRR*, abs/1808.01091, 2018.

[37] Qingyang Xi, Rachel M Bittner, Johan Pauwels, Xuzhou Ye, and Juan Pablo Bello. Guitarset: A dataset for guitar transcription. In *ISMIR*, pages 453–460, 2018.