Soundata: Reproducible use of audio datasets

Magdalena Fuentes [1]¶, **Genís Plaja-Roglans** [2], **Guillem Cortès-Sebastià** [2], **Tanmay Khandelwal** [1], **Marius Miron** [3], **Xavier Serra** [2], **Juan Pablo Bello** [1], and **Justin Salamon** [4]

1 New York University, New York, United States **2** Universitat Pompeu Fabra, Barcelona, Spain **3** Earth Species Project, Barcelona, Spain **4** Adobe Research, San Francisco, United States ¶ Corresponding author

Summary

Soundata is an open-source Python library for working with audio datasets in a programmatic and standardized way. It removes the need for writing custom loaders and improves reproducibility by providing tools to validate data against a canonical version. It speeds up research pipelines by allowing users to quickly download a dataset, validate that the dataset is complete and correct, and load it into memory in a standardized and reproducible way. It is designed to work with bioacoustics, environmental, urban, and spatial sound datasets; to be easy to use and easy to contribute to; and to increase reproducibility and standardize the usage of sound datasets in a flexible way.

Statement of need

As research pipelines become increasingly complex, it is key that their different components are reproducible. In recent years, the research community has made considerable efforts towards standardization and reproducibility, with modelling and evaluation libraries (Abadi et al., 2016; Mesaros et al., 2016; Pedregosa et al., 2011), open sourcing models (Ravanelli et al., 2021; Zinemanas et al., 2020), and data dissemination using resources such as Zenodo. However, discrepancies in the local version of the data and different practices in loading and parsing datasets can lead to considerable differences in performance results, which is misleading when comparing methods (Bittner et al., 2019). In addition, it is extremely inefficient to develop pipelines from scratch for loading and parsing a dataset for each researcher or team each time, and this increases the chances of bugs and differences that hinder reproducibility.

Soundata is based on and inspired by `Mirdata` (Bittner et al., 2019), the popular library for working with Music Information Research (MIR) datasets, and shares its goals and vision. However, in MIR, the aforementioned issues are exacerbated due to the intrinsic commercial nature of music data, since it is very difficult to get licenses to distribute music recordings openly. Since musical datasets are extremely complex compared to other audio datasets, using the same software package for handling music and other audio datasets would lead to a very complex, hard-to-manage repository, which would be difficult to scale. Instead, we introduce Soundata as a separate effort that specifically addresses the annotation types and formats required by communities like DCASE[1], which work with bioacoustics, environmental, urban, and spatial sound datasets.

There are other libraries that handle datasets like `Tensorflow` (Abadi et al., 2016) or `Tensorflow Datasets` (TensorFlow, 2019), `DCASE-models` (Zinemanas et al., 2020), and `HuggingFace Datasets` (Lhoest et al., 2021). But none of them serves as a stand-alone library that can

---

[1]https://dcase.community/

easily be plugged into different work pipelines, with different modeling software. Having a community-centric, open-source, audio-specialized library allows us greater flexibility to incorporate more audio-specific API functionalities and align our priorities with those of the audio community.

Soundata follows these design principles:

- **Easy to use**: Simplifies audio research pipelines considerably by having plug-and-play datasets in a standardized format.

- **Easy to contribute to:** Users do not need to go through all the source code to contribute. Soundata provides extensive documentation explaining how to contribute a new loader.

- **Increase reproducibility:** Provides a common framework for users to compare and validate their data. It also allows them to easily propagate dataset updates or fixes to the audio community, ensuring that methods are still comparable and users have the same up-to-date dataset versions. On that note, Soundata is designed to handle multiple versions of the same dataset, allowing transparent access to all versions of the dataset.

- **Standardize usage of sound datasets:** Standardizes common attributes of sound datasets such as audio or tags to simplify audio research pipelines, while preserving the idiosyncrasies of each dataset (e.g., if a dataset has 'non-standard' attributes, we include them as well).

Design Choices

Soundata has three main components, depicted in Figure 1: a core module that implements the generic functions used by all the data loaders (e.g., Dataset), a `utils` module with the main utility functions such as downloading and validating the data or converting to JAMS[2] format, and the dataset `loaders` containing dataset-specific code to load and parse each dataset in a standardized way. Following this design, when a new dataset requires a new functionality, it is added to the core module so it can be used for similar loaders added later on.
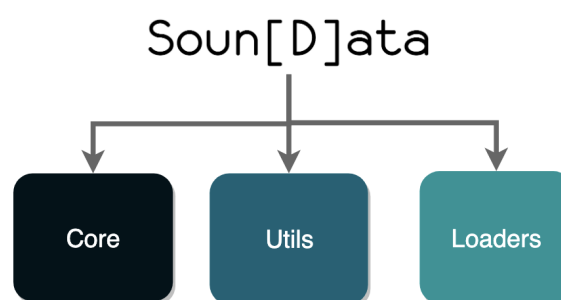
Figure 1: Soundata's main components.

Annotation Types

Annotation types in Soundata (see Figure 2) ensure compatibility with existing evaluation libraries from the DCASE community such as sed_eval, and are convertible to the JAMS format. These annotation types allow Soundata to support a wide range of audio research tasks, as shown in Figure 3. It currently includes three annotation types:

---

[2]https://github.com/marl/jams

- **Tags**: String labels with associated confidence values, spanning the full duration of the audio clip.
- **Events**: These annotations are for sound events with defined start times, end times, labels, and (optionally) confidence values.
- **Spatial Events**: Spatial Events extends Events introducing additional attributes such as geographical coordinates (latitude, longitude), altitude, direction (azimuth and elevation), and distance from reference points.
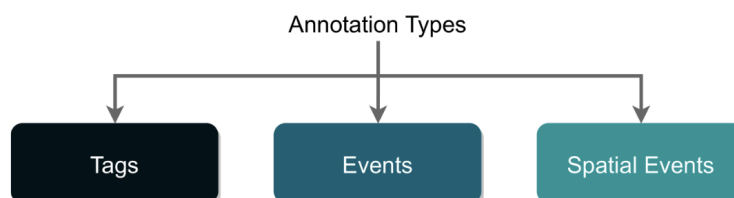
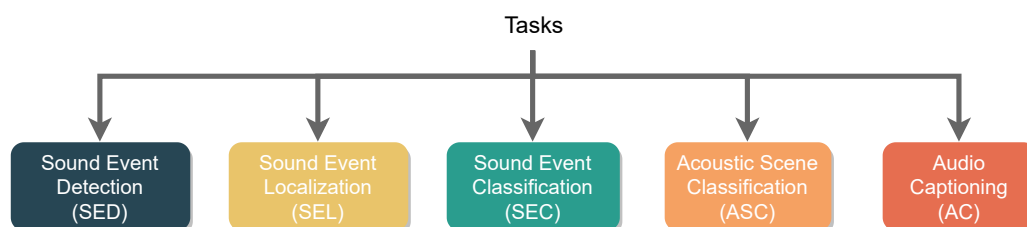**Figure 2:** Annotation types included in `Soundata`.

Figure 3: Audio tasks supported by `Soundata` as of today.

Example usage

`Soundata` is designed to be user-friendly, so that users can start working with audio datasets right away after following a few steps, as summarized in Figure 4.
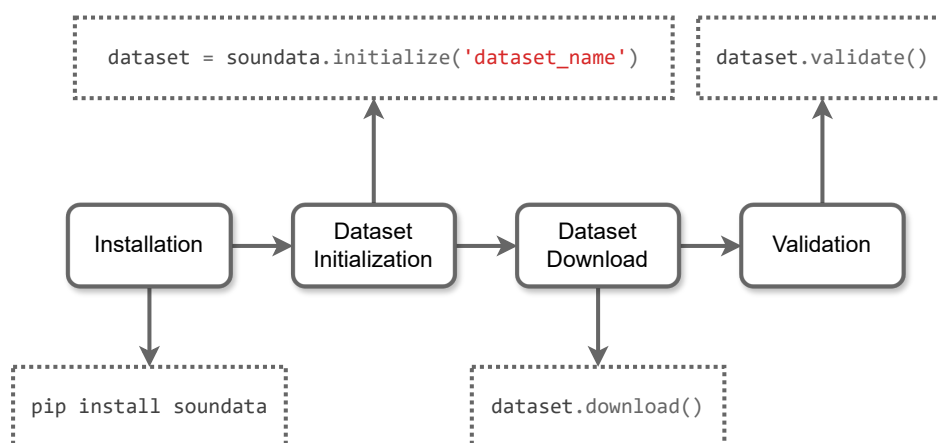
Figure 4: How to work with any supported dataset in `Soundata`.

Once the dataset is downloaded and validated, `Soundata` can be integrated into an audio research pipeline easily. The code in Figure 5 shows an example of how to get any SED dataset into a deep learning pipeline using `Soundata` and `Tensorflow`.

```
1   import soundata
2   import tensorflow as tf
3
4   def data_generator(dataset_name):
5       dataset = soundata.initialize(dataset_name)
6       dataset.download()  # Download dataset if needed
7       for clip_id in dataset.clip_ids:
8         clip = dataset.clip(clip_id)
9         # Assume sample rate consistency or handle as needed
10        audio_signal, _ = clip.audio
11        if clip.tags.labels:
12          label = clip.tags.labels[0]
13        else:
14          label = "Unknown"
15        yield audio_signal.astype("float32"), label
16
17  # Create a Tensorflow dataset
18  tf_dataset = tf.data.Dataset.from_generator(
19      lambda: data_generator("urbansound8k"),
20      output_types=(tf.float32, tf.string)
21  )
22
23  # Example: Iterate through the dataset
24  for audio, label in tf_dataset.take(1):
25      print("Audio Shape:", audio.shape)
26    print("Label:", label)
```

Figure 5: Soundata usage example. It shows an example of how to get any SED dataset into a deep learning pipeline.

Contributing

Contribution to Soundata is highly encouraged. To facilitate the process, Soundata provides an exhaustive contributing guide[3] available in the documentation with all the necessary information on how to contribute. The most common contribution in Soundata is the creation of new dataset loaders, as they play a crucial role in advancing Soundata's objective of accommodating as many datasets as possible. Figure 6 summarizes the process of creating a new loader.
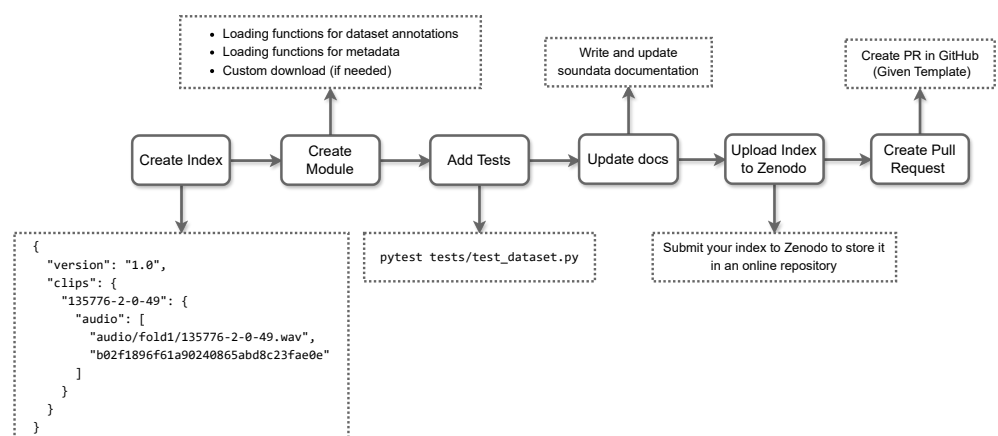
Figure 6: Steps for contributing a dataset loader to Soundata.

3https://soundata.readthedocs.io/en/latest/source/contributing.html

Acknowledgements

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 265–283. https://doi.org/10.48550/arXiv.1605.08695

Bittner, R. M., Fuentes, M., Rubinstein, D., Jansson, A., Choi, K., & Kell, T. (2019). Mirdata: Software for reproducible usage of datasets. *ISMIR*. https://doi.org/10.5281/zenodo.3527750

Lhoest, Q., Moral, A. V. del, Jernite, Y., Thakur, A., Platen, P. von, Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., & others. (2021). Datasets: A community library for natural language processing. *arXiv Preprint arXiv:2109.02846*. https://doi.org/10.48550/arXiv.2109.02846

Mesaros, A., Heittola, T., & Virtanen, T. (2016). Metrics for polyphonic sound event detection. *Applied Sciences*, *6*(6), 162. https://doi.org/10.3390/app6060162

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830. https://doi.org/10.48550/arXiv.1201.0490

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., … Bengio, Y. (2021). *SpeechBrain: A general-purpose speech toolkit*. https://doi.org/10.48550/arXiv.2106.04624

TensorFlow. (2019). *TensorFlow Datasets: A collection of ready-to-use datasets*. https://www.tensorflow.org/datasets.

Zinemanas, P., Hounie, I., Cancela, P., Font Corbera, F., Rocamora, M., & Serra, X. (2020). DCASE-models: A Python library for computational environmental sound analysis using deep-learning models. In N. Ono, N. Harada, Y. Kawaguchi, A. Mesaros, K. Imoto, Y. Koizumi, & T. Komatsu (Eds.), *Proceedings of the fifth workshop on detection and classification of acoustic scenes and events (DCASE 2020)*. Detection; Classication of Acoustic Scenes; Events (DCASE). https://doi.org/10.5281/zenodo.4061782