# audb - Sharing and Versioning of Audio and Annotation Data in Python

*Hagen Wierstorf [1], Johannes Wagner [1], Florian Eyben [1], Felix Burkhardt [1], Björn W. Schuller [1,2,3]*

[1] audEERING GmbH, Gilching, Germany
[2] Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany
[3] GLAM – Group on Language, Audio, & Music, Imperial College, UK
hwierstorf@audeering.com



## Abstract

Driven by the need for larger and more diverse datasets to pre-train and fine-tune increasingly complex machine learning models, the number of datasets is rapidly growing. audb is an open-source Python library that supports versioning and documentation of audio datasets. It aims to provide a standardised and simple user-interface to publish, maintain, and access the annotations and audio files of a dataset. To efficiently store the data on a server, audb automatically resolves dependencies between versions of a dataset and only uploads newly added or altered files when a new version is published. The library supports partial loading of a dataset and local caching for fast access. audb is a lightweight library and can be interfaced from any machine learning library. It supports the management of datasets on a single PC, within a university or company, or within a whole research community.

## 1. Introduction

To foster progress in automatic speech emotion recognition and related learning tasks, it is crucial to have a quick and easy way of accessing an ensemble of datasets for training and evaluation [1]. This requires that the datasets have a unique identifier, are versioned, can be shared and combined, are documented in a standardised way [2], and can be accessed from a common user interface.

This paper presents *audb*, a Python library to publish, maintain, and access labelled or unlabelled audio data in machine learning pipelines. It also supports audio tracks embedded in video files. It can load a dataset by name and version from different repositories. The dataset is then provided in a well defined specification (*audformat*[1]), and its audio data can be resampled, remixed, or converted to the desired format. A caching mechanism guarantees quick access. A dataset consists of a root folder with a header file and multiple table files holding metadata and annotations, and the referenced audio files, usually organised into sub-directories. The Python library *audinterface*[2] provides an interface to read and process the audio data of one or more datasets.

*audb* is under continuous development and has been used to publish and maintain 840 datasets and versions since 4 years inside audEERING. It is released open-source since 2021 under an MIT license, available via PyPI[3] and Github,[4] and the documentation is hosted on the project website.[5]

[1] https://audeering.github.io/audformat/
[2] https://audeering.github.io/audinterface/
[3] https://pypi.org/project/audb/
[4] https://github.com/audeering/audb/
[5] https://audeering.github.io/audb/

## 2. Related Work

With the introduction of Git in 2005 and platforms like Github in 2008 for development and sharing of code, it became obvious that no convenient solution for audio data management and sharing existed within the research community. In 2014, Git Large File Storage[6] was released and established as the standard way of including binary files in git repositories based on similar ideas like git-media[7] which existed already since 2009. With this approach, it became possible to have git repositories that provide versioning of data and track authorship of certain changes to the data. As Git Large File Storage did not focus on a particular kind of binary data to be versioned Data Version Control[8] evolved since 2017 with a focus on versioning data, machine learning models, and experiments to foster reproducibility [3, 4, 5].

In parallel, the problem of sharing large amount of research data was tackled by approaches like Zenodo established in 2013 [6]. Zenodo allows researchers to upload datasets and provides a digital object identifier [7] to make datasets easier to cite and provide long-term access to them [8]. Access to shared data can be improved if the data and its corresponding metadata or annotations are also provided in a standardised way. One successful example from the audio community is the Spatially Oriented Format for Acoustics (SOFA) format for impulse responses [9].

Recently, different audio communities have addressed the problem of reproducibility with open-source toolkits, which help to re-run experiments and access related datasets. Bittner *et al.* [10] introduced a Python library to load and manage annotations for Music Information Retrieval (MIR) datasets, which was later extended for more general audio datasets [11]. The audio source separation community develops the Asteroid toolkit which can access relevant datasets [12]. More general toolkits like SpeechBrain [13], TensorFlow [14], or PyTorch [15] include handling of data, but do not focus on data versioning and management. The Hugging Face *Datasets* [16] library extends the dataset handling from TensorFlow and makes it independent of any machine learning library. It provides access to datasets for natural language processing, but also computer vision, and audio. *Datasets* can efficiently handle very huge datasets by streaming the data and loading it only partially into memory. On the Hugging Face Hub[9], it provides repositories for datasets in which the versioning is handled by Git and Git Large File Storage. As *Datasets* addresses data management in a similar way to *audb*, Section 4 will compare them in more depth.

[6] https://git-lfs.com
[7] https://github.com/alebedev/git-media
[8] https://dvc.org
[9] https://huggingface.co/datasets

Table 1: *Default metadata entries in the header of a dataset. If needed, the list can be extended by custom fields.*

| Field | Mandatory | Description |
|---|---|---|
| name | yes | name of dataset |
| source | yes | original source, e. g., URL |
| usage | yes | data usage, e. g., research |
| author | | author(s) |
| description | | long description |
| expires | | expiration date if applicable |
| languages | | included languages |
| license | | license |
| organisation | | organisation |

## 3. Library Overview and Design

The most important functionality of the library is to load a dataset and access its annotations and files. The following example loads version 1.3.0 of the emodb dataset [17] and returns the file names and corresponding annotations stored in a table with the name 'emotion' as a pandas dataframe:

```
db = audb.load("emodb", version="1.3.0")
df = db["emotion"].get()
```

A complete list of all available functions and classes with examples is provided with the *audb* API documentation.[10]

### 3.1. Annotations, Metadata and Header

Annotations are stored as columns in tables, which are represented in a human readable way by CSV files named 'db.<table id>.csv', and cached as pickle files for faster access. Each table and column is identified by a unique ID. The rows in the tables are associated with audio files or segments of audio files, which define the index of the table. A *filewise* index is used to reference files as a whole:

```
file,emotion
a.wav,happy
b.wav,angry
```

Or if segments should be referenced, a *segmented* index with additional start and end times is used:

```
file,start,endemotion
c.wav,0 days 00:00:01.0,0 days 00:00:03.3,happy
c.wav,0 days 00:00:03.5,0 days 00:00:07.8,angry
```

Annotations that are not attached to a file can be organised in *misc tables*, which support custom indices. e. g., the following table stores age and gender of the speakers in the dataset:

```
speaker,age,gender
spk0,29,female
spk1,93,male
```

It is possible to restrict the values in a column of a (misc) table to a certain data type or range (bool, date, float, integer, object, string, time) by assigning it to a scheme. For example, a column with annotations of emotion can be restricted to a set of labels like 'happy', 'angry', 'neutral'. This information is stored in the header of a dataset.

In addition, the header lists information about attachments, tables, columns, raters, and splits. And it stores metadata about the dataset, which is summarised Table 1. The header is saved in a YAML file, which is located in the root folder of the dataset.

[10] https://audeering.github.io/audb/api/audb.html

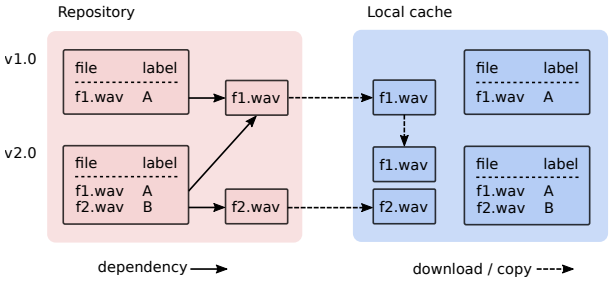



Figure 1: *A dependency system ensures that only new or altered files are uploaded to the server for new dataset versions. For instance, in v2.0, a dependency to 'f1.wav' from v1.0 is set (left part). When the dataset is loaded, references are resolved and a self-contained copy of the dataset is created in the cache. If possible, files are retrieved from other versions of the dataset that exist in the cache. For instance, when loading v2.0 the file 'f1.wav' is copied from v1.0 (right part).*

### 3.2. Repositories and Backends

*audb* can host datasets in one or more repositories, which can be distributed over different backends. Currently, *audb* supports a local file system, and an Artifactory instance as backend. But it is also possible to implement custom backends and register them with *audb*.

The repository of a dataset is defined by a name, a host, and a backend, and can be obtained via:

```
repo = audb.repository("emodb", version="1.3.0")
repo.name
repo.host
repo.backend
```

### 3.3. Publication and Versioning

A new dataset is published from a root folder, which contains its header and tables, as well as, the referenced audio files, which are possibly organised into sub-folders. For instance, consider a dataset with two audio files and a table with ID 'emotion', stored in a folder 'dataset':

```
dataset/
    audio/
        a.wav
        b.wav
    db.emotion.csv
    db.yaml
```

The dataset can then be published as version 1.0.0 to some repository with:

```
audb.publish("./dataset", "1.0.0", repository)
```

The dataset header, table, and audio files are uploaded as individual ZIP files to the repository. In addition, a dependency table is created, which holds metadata about audio files (e. g., sampling rate) and records for every table and audio file in which version of the dataset it is stored. The entries in the dependency table are also available to the user, e. g., compare the sampling rate entry in the model card in Figure 2.

To publish a new version of a dataset, a user downloads header and tables of a previous version, and optionally also audio files if she plans to replace them. Now, existing files can be deleted or modified, and new files can be added. Afterwards, the dataset can be published under a new version. During publication *audb* automatically identifies the changes and uploads

only the new or altered files. For the remaining files, a dependency to the version in which it was last modified is set (see Figure 1).

For well established datasets or datasets that grow over time, we recommend to version the scripts that publish the dataset on a service like Github. This allows users to open issues or create pull requests to fix errors in the dataset. An example of such a dataset project for the emodb dataset [17] can be found at https://github.com/audeering/emodb.

Since *audb* handles the versioning and can automatically detect changes made to a dataset, it is possible to fully automate the publishing process. This allows it to directly integrate the data publishing into annotation or data collection tools.

### 3.4. Flavours

When loading a dataset, by default, the original audio files are retrieved. However, *audb* offers the option to request a dataset in a specific flavour. In that case the audio files are converted to the same format with a specific bit depth, sampling rate, channel selection, and mix-down. For each flavour, a separate cache folder is used, i. e., the same dataset may be available in different formats. In a machine learning pipeline, flavours can be used to ensure that audio files stemming from different datasets are in the same format, e. g., share a sampling rate of 8000 Hz:

```
db = audb.load("emodb", sampling_rate=8000)
```

### 3.5. Partial Loading

To speed up loading, it is possible to request only specific parts of a dataset. For example, the header and tables of a dataset can be loaded without audio files:

```
db = audb.load("emodb", only_metadata=True)
```

Or specific tables can be loaded, which will only load audio files referenced in those tables:

```
db = audb.load("emodb", tables="emotion")
```

Or specific audio files can be loaded, which will automatically remove other entries from the tables:

```
db = audb.load("emodb", media="wav/03a01Fa.wav")
```

### 3.6. Caching

When a dataset is loaded, *audb* figures out missing tables and audio files, and either copies them from an already cached version of the dataset or, if that is not possible, downloads them from the server (see Figure 1). If the dataset is completely cached, loading works without an internet connection. Inside the cache a folder is created for every version and flavour of a dataset. Dependencies to earlier versions are automatically resolved so that the folder in which the dataset is stored contains all files. This consumes more space, but has the advantage that the dataset is self-contained and can be shipped as is and directly loaded with *audformat*. Tables are cached as CSV files, and in addition pickled for fast reading.

### 3.7. Removing Audio Files From All Versions

Audio recordings may contain sensitive information. Therefore, *audb* offers the option to remove specific audio files from all published versions. This goes beyond dropping files with a new dataset version as discussed in Section 3.3, which does not remove files from previous versions. This can result in non-reproducibility of some results, but avoids completely removing affected versions of the dataset.

## 4. Comparison with Hugging Face Datasets

The Hugging Face Hub provides data repositories to publish datasets with Hugging Face *Datasets*. A data repository contains documentation of the dataset and a Git repository with Git Large File Storage support to version the data. A data repository can contain a so called loading script, which *Datasets* executes when loading the data. This allows *Datasets* to download data from external sources and easily incorporate public datasets already stored somewhere, e. g., on Zenodo. As a consequence of this approach, *Datasets* lacks information which files persist between versions of a dataset and therefore all data (again) has to be downloaded when a new version of a dataset is requested. In contrast, *audb* does not support linking external sources as all audio files must be part of the repository. This, however, enables *audb* to store and load the data more efficiently since the same file can be shared across versions. Another disadvantage of loading datasets with a script is that rolling out a dataset can be slow, as it might require parsing a million lines of annotations first and convert them from row to column representation.

*audb* can download single audio files from a dataset, whereas with *Datasets*, this is only possible if the creator of a dataset puts each audio file into a single archive with the name of the audio file so that it can be addressed during download. However, the common approach with *Datasets* is to not publish individual audio files, but bundle them into few large splits (e. g., train, dev, test) as in the case for Librispeech [18].[11]

*Datasets* scales to very large datasets as its data loading is based in Apache Arrow[12] and allows it to load datasets only partially into memory and to stream datasets when downloading them. *audb* always has to load whole tables into memory. It offers two strategies for avoiding high memory consumption: splitting into smaller tables and using partial loading (see Section 3.5). An advantage of *audb* is that it uses pickled files, which read faster than Apache Arrow files.

*Datasets* does not support organising annotations into different tables, or referencing the same audio files or parts of it multiple times. Each data point that is returned contains the actual audio signal, a link to the corresponding audio file and associated labels. Whereas in *audb*, there is only a loose connection between audio files and annotations. This means there is exactly one copy of an audio file, even when it is referenced from different tables or different, possibly overlapping segmentation exist. It further allows mapping annotations from one table to another. For example, consider the following three tables:

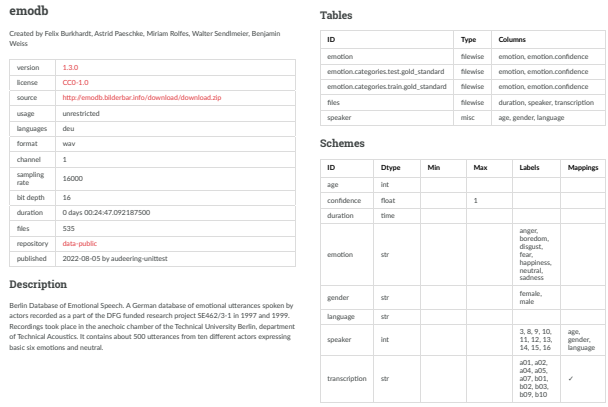
```
# ID: speakers
speaker,age
spk01,19
spk02,21

# ID: files
file,speaker
a.wav,spk01
b.wav,spk02

# ID: emotion
file,start,end,emotion
a.wav,0,0 days 00:00:01,happy
a.wav,0,0 days 00:00:02,calm
```

[11] https://huggingface.co/datasets/librispeech_asr

[12] https://github.com/apache/arrow



**emodb**

Created by Felix Burkhardt, Astrid Paeschke, Miriam Rolfes, Walter Sendlmeier, Benjamin Weiss

| | |
|---|---|
| version | 1.3.0 |
| license | CC0-1.0 |
| source | http://emodb.bilderbar.info/download/download.zip |
| usage | unrestricted |
| languages | deu |
| format | wav |
| channel | 1 |
| sampling rate | 16000 |
| bit depth | 16 |
| duration | 0 days 00:24:47.092187500 |
| files | 535 |
| repository | data-public |
| published | 2022-08-05 by audeering-unittest |

**Description**

Berlin Database of Emotional Speech. A German database of emotional utterances spoken by actors recorded as a part of the DFG funded research project SE462/3-1 in 1997 and 1999. Recordings took place in the anechoic chamber of the Technical University Berlin, department of Technical Acoustics. It contains about 500 utterances from ten different actors expressing basic six emotions and neutral.

**Tables**

| ID | Type | Columns |
|---|---|---|
| emotion | filewise | emotion, emotion.confidence |
| emotion.categories.test.gold_standard | filewise | emotion, emotion.confidence |
| emotion.categories.train.gold_standard | filewise | emotion, emotion.confidence |
| files | filewise | duration, speaker, transcription |
| speaker | misc | age, gender, language |

**Schemes**

| ID | Dtype | Min | Max | Labels | Mappings |
|---|---|---|---|---|---|
| age | int | | | | |
| confidence | float | | 1 | | |
| duration | time | | | | |
| emotion | str | | | anger, boredom, disgust, fear, happiness, neutral, sadness | |
| gender | str | | | female, male | |
| language | str | | | | |
| speaker | int | | | 3, 8, 9, 10, 11, 12, 13, 14, 15, 16 | age, gender, language |
| transcription | str | | | a01, a02, a04, a05, a07, b01, b02, b03, b09, b10 | ✓ |

Figure 2: *Excerpt of the data card for emodb [17]. It includes a description of the dataset and metadata like author and license, and lists the tables, columns and schemes in the dataset.*

If the 'speakers' table is assigned as scheme to the 'speaker' column of the 'files' table, its labels can be mapped to the values of a column in the 'speakers' table, e. g., 'age'. And it can be requested using the segmentation from the 'emotion' table as index:

```
db["files"]["speaker"].get(
    index=db["emotion"].index,
    map="age",
)
```

The result is a segmented table with the age of the speakers:

```
file,start,end,age
a.wav,0,0 days 00:00:01,19
a.wav,0,0 days 00:00:02,19
```

# 5. Use Cases

## 5.1. Browsing and Searching Datasets

*audb* provides the possibility to list available datasets.

```
datasets = audb.available(only_latest=True)
```

The results can be filtered for datasets that have a scheme 'emotion':

```
# Create scheme lookup dictionary
schemes = {}
for name, version in datasets.version.items():
    schemes[name] = list(
        audb.info.schemes(name, version=version)
    )
# Search for datasets with scheme "emotion"
emotional_datasets = [
    name for name in schemes
    if "emotion" in schemes[name]
]
```

Since metadata and annotations are provided in a well defined format, it is possible to automatically create documentation in form of data cards or datasheets [2]. Figure 2 shows an example data card for the emodb dataset [17]. It summarises the most important facts in a tabular form, provides a long description of the dataset together with an audio example and lists available tables, columns, and schemes. Data cards for all datasets available with the default public repositories of *audb* are available at https://audeering.github.io/datasets/.

## 5.2. Fine-tuning a Model for Emotion Recognition

With the emergence of foundation models [19] pre-trained on large amounts of data, it is nowadays a common task in the paralinguistic community to fine-tune generic models to a specific problem. The following example shows how this can be easily achieved using *audb* and *audinterface*.

Assume we have a callable model that converts an audio signal into a compact feature representation (embeddings). We first create an interface for it:

```
feature_extractor = audinterface.Process(
    process_func=model,
    num_workers=4,
)
```

Then, we load a dataset and convert it into a single feature matrix on which we train a linear model that predicts the emotional content of the input signal:

```
db = audb.load("emodb", version="1.3.0")
labels = db["emotion"]["emotion"].get()
features = feature_extractor.process_index(
    labels.index
)
```

For a full example see https://github.com/audeering/w2v2-how-to/blob/main/notebook.ipynb.

## 5.3. Publishing new dataset splits

Datasets of emotional speech might be published without an official train, dev, test split like emodb [17] or IEMOCAP [20]. Other datasets such as CommonVoice [21] or VoxCeleb [22] might miss splits for tasks not considered originally when collecting the data, e. g., age prediction. *audb* makes it easy to add new tables and splits to a dataset and publish a new version, encouraging other researchers to reuse them. For example, with version 1.2.0 we added a train-test split to the emodb dataset:

```
audb.load_to("./db", "emodb", version="1.1.1")
# Add new splits to db
audb.publish("./db", "1.2.0", repository)
```

# 6. Conclusion

The recent success of foundation models [19] in the paralinguistic and audio community has raised the need for a large number of diverse datasets to train and evaluate models fine-tuned to a specific task [23]. *audb* is a lightweight, yet powerful Python library to publish, maintain and access audio datasets and their annotations. Its highlights are: a built-in versioning system, an automated workflow to publish and update datasets locally or on a remote server, and sharing datasets to specific end-users or communities. We published a selection of publicly available datasets in a public repository. The repository is pre-configured in *audb* and the datasets can be directly accessed. For a list of available datasets please visit anonymised.

# 7. Acknowledgements

The authors would like to thank Baha Eddine Abrougui, Christian Geng, Stephan Huber, Andreas Triantafyllopoulos, Damiano Zanardo for their contributions, and the pandas community for providing the basis for audb. This research has been partly funded by the European EASIER (Grant Agreement number: 101016982) and by the European SHIFT project (Grant Agreement number: 101060660).

# 8. References


[1] N. Scheidwasser-Clow, M. Kegler, P. Beckmann, and M. Cernak, "Serab: A multi-lingual benchmark for speech emotion recognition," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7697–7701. DOI: 10 . 1109 / ICASSP43922 . 2022 . 9747348.

[2] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. Iii, and K. Crawford, "Datasheets for datasets," *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, 2021. DOI: 10.1145/3458723.

[3] V. C. Stodden and Yale Roundtable Participants, "Reproducible research: Addressing the need for data and code sharing in computational science," *IEEE Computing in Science and Engineering*, vol. 12, no. 5, pp. 8–13, 2010. DOI: 10.1109/MCSE.2010.113.

[4] B. K. Olorisade, P. Brereton, and P. Andras, "Reproducibility in machine learning-based studies: An example of text mining," in *Proceedings of the Reproducibility in ML Workshop at the $34^{th}$ International Conference on Machine Learning, ICML*, vol. 70, Sydney, NSW, Australia, 2017.

[5] M. Pawlik, T. Hütter, D. Kocher, W. Mann, and N. Augsten, "A link is not enough–reproducibility of data," *Datenbank-Spektrum*, vol. 19, pp. 107–115, 2019. DOI: 10.1007/s13222-019-00317-8.

[6] A. Purcell, *CERN and OpenAIREplus launch new european research repository*, https://sciencenode.org/feature/cern-and-openaireplus-launch-new-european-research-repository.php, Accessed: 2023-02-21, 2013.

[7] *ISO 26324:2022(en), information and documentation – digital object identifier system*, https://www.iso.org/obp/ui/#iso:std:iso:26324:ed-2:v1:en, Accessed: 2023-02-21, International Organization for Standardization, 2022.

[8] P. Herterich and S. Dallmeier-Tiessen, "Data citation services in the high-energy physics community," *D-Lib Magazine*, vol. 22, no. 1/2, 2016. DOI: 10.1045/january2016-herterich.

[9] P. Majdak, Y. Iwaya, T. Carpentier, R. Nicol, M. Parmentier, A. Roginska, Y. Suzuki, K. Watanabe, H. Wierstorf, H. Ziegelwanger, and M. Noisternig, "Spatially oriented format for acoustics: A data exchange format representing head-related transfer functions," in *Proceedings of the $134^{th}$ Convention of the Audio Engineering Society*, Roma, Italy: Audio Engineering Society, 2013.

[10] R. M. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi, and T. Kell, "Mirdata: Software for reproducible usage of datasets," in *Proceedings of the $20^{th}$ International Society for Music Information Retrieval Conference, ISMIR*, Delft, The Netherlands, 2019, pp. 99–106, ISBN: 9781732729919.

[11] M. Fuentes, J. Salamon, P. Zinemanas, M. Rocamora, G. Paja, I. R. Román, M. Miron, X. Serra, and J. P. Bello, "Soundata: A python library for reproducible use of audio datasets," *arXiv preprint arXiv:2109.12690*, 2021.

[12] M. Pariente, S. Cornell, J. Cosentino, S. Sivasankaran, E. Tzinis, J. Heitkaemper, M. Olvera, F.-R. Stöter, M. Hu, J. M. Martín-Doñas, D. Ditter, A. Frank, A. Deleforge, and E. Vincent, "Asteroid: The PyTorch-based audio source separation toolkit for researchers," in *Proceedings of the $21^{st}$ Annual Conference of the International Speech Communication Association, INTERSPEECH 2020*, Virtual Event, Shanghai, China: International Speech Communication Association, 2020, pp. 2637–2641. DOI: 10.21437/Interspeech.2020-1673.

[13] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, *et al.*, "Speechbrain: A general-purpose speech toolkit," *arXiv preprint arXiv:2106.04624*, 2021.

[14] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the $12^{th}$ USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, Savannah, GA, USA: USENIX Association, 2016, pp. 265–283.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of the $33^{rd}$ International Conference on Neural Information Processing Systems (NIPS '19)*, Vancouver, BC, Canada: Neural Information Processing Systems Foundation, 2019, pp. 8026–8037.

[16] Q. Lhoest, A. Villanova del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall, J. Davison, M. Šaško, G. Chhablani, B. Malik, S. Brandeis, T. Le Scao, V. Sanh, C. Xu, N. Patry, A. McMillan-Major, P. Schmid, S. Gugger, C. Delangue, T. Matussière, L. Debut, S. Bekman, P. Cistac, T. Goehringer, V. Mustar, F. Lagunas, A. Rush, and T. Wolf, "Datasets: A community library for natural language processing," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 175–184. DOI: 10.18653/v1/2021.emnlp-demo.21.

[17] F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier, and B. Weiss, "A database of German emotional speech," in *Proceedings of the $9^{th}$ European Conference on Speech Communication and Technology, Eurospeech, INTERSPEECH 2005*, Lisbon, Portugal: International Speech Communication Association, 2005, pp. 1517–1520, ISBN: 9781604234480.

[18] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, South Brisbane, QLD, Australia, 2015, pp. 5206–5210. DOI: 10 . 1109 / ICASSP . 2015 . 7178964.

[19] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[20] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "IEMOCAP: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, no. 4, pp. 335–359, 2008. DOI: 10.1007/s10579-008-9076-6.

[21] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," in *Proceeding of the $12^{th}$ International Conference on Language Resources and Evaluation, LREC*, Marseille, France: European Language Resources Association (ELRA), 2020, pp. 4218–4222.

[22] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Proceedings of the $18^{th}$ Annual Conference of the International Speech Communication Association, INTERSPEECH 2017*, Stockholm, Sweden: International Speech Communication Association, 2017, pp. 2616–2620. DOI: 10.21437/Interspeech.2017-950.

[23] J. Turian, J. Shier, H. R. Khan, B. Raj, B. W. Schuller, C. J. Steinmetz, C. Malloy, G. Tzanetakis, G. Velarde, K. McNally, M. Henry, N. Pinto, C. Noufi, C. Clough, D. Herremans, E. Fonseca, J. Engel, J. Salamon, P. Esling, P. Manocha, S. Watanabe, Z. Jin, and Y. Bisk, "HEAR: Holistic Evaluation of Audio Representations," in *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, D. Kiela, M. Ciccone, and B. Caputo, Eds., vol. 176, PMLR, 2022, pp. 125–145.