forum acusticum 2026
8-12 September · Graz, Austria

HRTFPYKIT: A PYTHON LIBRARY FOR HRTF RESEARCH WORKFLOWS

Ariel Alvarez-Martinez[1], Jose J. Lopez[2]

[1] *Independent researcher, Spain.*
[2] *Institute of Telecommunications and Multimedia Applications, Universitat Politècnica de València, Spain.*

Abstract

Head-related transfer functions (HRTFs) describe how the human head, torso, and pinnae filter sound arriving from different directions. They are unique to each listener and are an important part of binaural rendering, spatial audio, and virtual reality systems. HRTF research involves not only acoustic analysis but also the management of Spatially Oriented Format for Acoustics (SOFA) files, public datasets, and reproducible data pipelines for analysis and deep learning model training. In practice, this work often relies on custom scripts combined with existing software packages, resulting in heterogeneous workflows that make experiments harder to reproduce or extend. These challenges motivated the development of hrtfpykit, a Python library for HRTF research workflows. It includes a `sofa` layer for reading, writing, and editing SOFA files. The `hrtf` layer represents HRTF data as a unified object containing synchronized impulse-response and transfer-function representations and supports acoustic transformations and spatial and spectral selection. The `plots` layer provides visualization tools for HRTF inspection and comparison in the time and frequency domains. Finally, the `datasets` layer standardizes access to public HRTF datasets through declarative specifications and configurable transformations, enabling reproducible pipelines for training deep learning models, particularly for HRTF individualization.

Keywords: *hrtf, sofa, research workflows, datasets*

1. Introduction

Head-related transfer functions (HRTFs) describe the acoustic filtering produced by the human head, torso, and pinnae as sound propagates from a source to the listener [1]. They constitute a fundamental component of binaural rendering and are widely used in spatial audio, virtual and augmented reality, and auditory perception research. Since HRTFs are highly dependent on an individual's anatomy, anthropometric differences among listeners lead to different HRTF spectra. Consequently, using individualized rather than generic HRTFs can improve sound localization accuracy, reduce front-back confusions, enhance externalization, and support a more realistic spatial listening experience [2], [3].

Over the years, several software tools have been developed to support HRTF research. The SOFA Toolbox [4] and Sofar [5] provide interfaces for working with the Spatially Oriented Format for Acoustics (SOFA), a standardized representation for spatial acoustic data and associated metadata [6]. Hartufo [7] supports access to public HRTF datasets, while the Spatial Audio Metrics (SAM) toolbox [8] provides methods for spatial-audio analysis. Although these tools provide valuable functionality, they generally focus on specific stages of the research process. As a result, working across dataset access, SOFA manipulation, acoustic processing, visualization, and data preparation often requires combining multiple packages with custom scripts.

This fragmentation is particularly evident in recent deep-learning workflows for HRTF individualization. Public HRTF datasets must be downloaded, standardized, transformed, and organized into reproducible training and evaluation pipelines. Measured HRTFs are often paired with non-acoustic listener information used as model inputs, including anthropometric measurements, ear images, and geometric information from 3D meshes [9], [10], [11], [12], [13]. Before training a model, data from different resources must be matched by listener, standardized into compatible representations, consistently preprocessed, inspected visually, and split into training, validation, and test sets. When these steps are reimplemented separately for each project, much of the research effort shifts

***Corresponding author**: aalvmarprojects@gmail.com.*

away from acoustic and perceptual questions toward dataset handling and preparation, making workflows harder to reproduce and extend.

To address these challenges, we present `hrtfpykit`[1], an open-source Python library for HRTF research workflows. The library brings together SOFA file handling, acoustic processing, visualization, access to public HRTF datasets, and reproducible data pipelines. We provide extensive documentation for `hrtfpykit`, including tutorials that explain its main components and make the library easier to adopt.

The remainder of this paper is organized as follows. Section 2 presents the library's design principles. Section 3 describes its architecture and main components. Section 4 illustrates how these components are combined to support common HRTF research tasks. Section 5 summarizes the main contributions and outlines future directions.

2. Design principles

The design of `hrtfpykit` is guided by four main principles that support reproducible HRTF research.

2.1. Pythonic

Researchers working with HRTFs often rely on Python and its scientific ecosystem. `hrtfpykit` follows established Python conventions and offers consistent interfaces for loading, selecting, transforming, and combining data. It also integrates naturally with common libraries for numerical computing, visualization, and deep learning.

2.2. Research orientation

`hrtfpykit` aims to make common HRTF research tasks as direct and productive as possible. The complexity involved in opening SOFA files, creating time- and frequency-domain representations, generating plots, downloading resources, and constructing data pipelines is handled internally through clear interfaces.

2.3. Modularity

SOFA handling, HRTF representation, visualization, and dataset construction are implemented as separate but connected modules. Each module has a defined responsibility and communicates with the others. This separation reduces coupling, simplifies testing and debugging, and allows individual parts of the library to evolve without requiring unnecessary changes across the entire codebase.

2.4. Reproducibility

HRTF experiments involve a series of decisions about resource selection, acoustic representation, preprocessing, spatial selection, transformations, and dataset splitting. In `hrtfpykit`, these choices are expressed explicitly using classes, methods, specifications, configurations, and reusable transforms. This makes each data pipeline easier to inspect, reproduce, and apply consistently across subjects, datasets, and experiments.

[1]Public code and documentation are available at: `https://github.com/ArielAlvarez-Martinez/hrtfpykit` `https://hrtfpykit.readthedocs.io/`

3. Architecture

The public API of `hrtfpykit` is organized around four layers: `sofa`, `hrtf`, `plots`, and `datasets`, as shown in Figure 1. The `sofa` layer exposes the structure, variables, attributes, metadata, and conventions of SOFA files and provides the basis for acoustic data access. The `hrtf` abstraction builds on this representation by combining impulse responses, transfer functions, and source positions within a common object on which selections, transformations, and metrics can be applied. The `plots` module operates on HRTF objects and represents their current state through time, frequency, spatial, and comparison plots. The `datasets` layer manages public and custom resources, interprets declarative specifications, applies configurable transformations, and constructs map-style datasets for analysis and model training. These layers have separate responsibilities but interact through explicit data dependencies, illustrated by the dotted connections in Figure 1. The following subsections describe the role and main capabilities of each layer.

3.1. `sofa`

The `sofa` layer provides the low-level interface to SOFA files in `hrtfpykit`. It represents a SOFA file as a `SOFA` object and gives access to the structural elements of the file: dimensions, variables, global attributes, and variable attributes. These elements describe the organization of the acoustic data, the coordinate systems, the source and receiver positions, the sampling information, and the metadata required by the declared SOFA convention.

Although the higher layers of `hrtfpykit` focus on HRTF research, the `sofa` layer is not limited to HRTF-specific files. It can be used directly with SOFA files that follow the standardized SOFA 2.1 conventions, including general FIR and transfer-function data, free-field HRTFs, spatial room impulse responses, directivity data, and headphone impulse responses [14]. In this sense, the layer provides a general SOFA interface on which the HRTF-specific abstractions of the library are built.

The layer supports inspection, validation, editing, cloning, and saving of SOFA files. Validation utilities check the declared convention, required variables and attributes, dimension consistency, and custom fields. Additional security checks inspect file handling conditions and search for external links or suspicious attribute contents before a file is used. These operations keep file handling, metadata inspection, and convention checking close to the SOFA representation while providing a stable basis for the rest of the library.

3.2. `hrtf`

The `hrtf` layer provides the central acoustic abstraction of the library. It represents an HRTF as an object

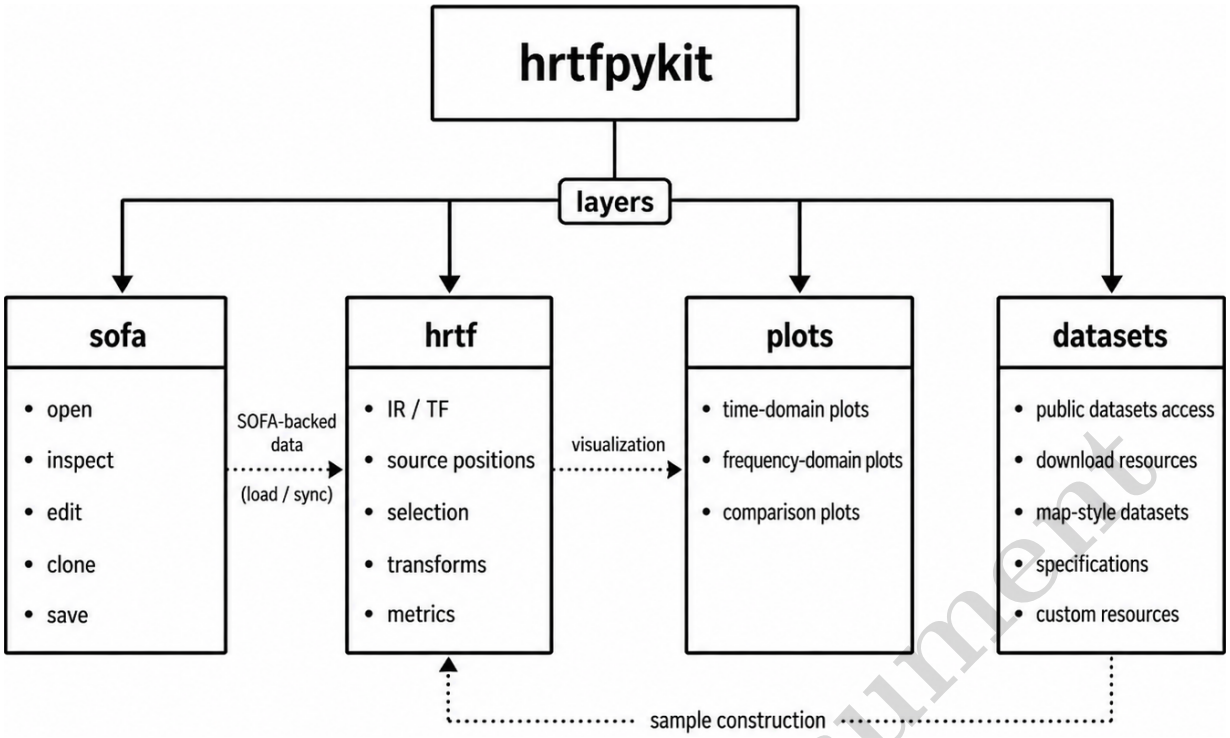

Figure 1: Overview of the hrtfpykit architecture.

that keeps the time-domain representation, frequency-domain representation, and source positions aligned. Files containing impulse responses are converted to transfer functions when loaded and, conversely, files containing transfer functions are converted to impulse responses. This gives the researcher access to both representations from the same object, independently of the SOFA convention used by the input file.

The object stores the acoustic data separately from the underlying SOFA representation. Selections and transformations are applied first to the in-memory HRTF object, and the SOFA file is updated only when the user explicitly synchronizes or saves the result. This separation makes it possible to inspect, subset, transform, compare, and export HRTFs without writing intermediate processing steps back to the original file.

Spatial information is handled through the source-position representation associated with the HRTF object. The layer supports coordinate conversion, named directions, plane queries, and spatial selections, so that acoustic operations can be applied to complete HRTFs or to selected directions. It also provides access to time- and frequency-domain descriptors such as impulse-response length, duration, frequency bins, magnitude, phase, real and imaginary components, as well as metrics for binaural cues and HRTF comparison, including interaural time difference (ITD), interaural level difference (ILD) [15], and log spectral distortion (LSD) [9].

Transformations include common time- and frequency-domain operations, such as windowing, cropping, padding, resampling, filtering, gain adjustment, phase or magnitude modification, and conversion to minimum-phase, common-transfer-function (CTF), and directional-transfer-function (DTF) representations [16].

3.3. `plots`

The `plots` layer makes the state of an HRTF object visible without adding plotting logic to the core acoustic abstraction. Plotting functions receive one or more HRTF objects and read their current impulse-response, transfer-function, and source-position representations. As a result, figures reflect the selections, transformations, domain conversions, and source-grid operations already applied to the object.

The layer provides plots for inspecting individual HRTFs in the time, frequency, and spatial domains, including impulse-response amplitudes, energy-time curves, magnitude spectra, spectrum planes, elevation spectra, source grids, spatial planes, and binaural cue plots. It also supports comparison figures across several HRTF objects, including magnitude and amplitude comparisons, ITD and ILD comparisons, difference plots, and HRTF difference metrics. These utilities provide a consistent visual interface for checking acoustic data, transformations, and model inputs before they are used in later analysis or training.

3.4. `datasets`

The `datasets` layer allows users to access resources from public HRTF datasets such as ARI, HUTUBS, and SONICOM [17], [18], [19]. Figure 2 summarizes its main steps: selecting resource families, obtaining or providing files, declaring specifications, and build-

ing the dataset. Public HRTF datasets may provide HRTF SOFA files, anthropometric information, 3D meshes, images, and metadata. `hrtfpykit` handles dataset-specific rules internally, including subject identifiers, folder layouts, resource variants, downloadable resource groups, exclusions, and local resource summaries.

Resource acquisition is independent of dataset construction. Users explicitly select the resource families and server used to download files. These files do not have to match the resources later used to build a dataset. A researcher may retrieve only a subset of a dataset, reuse files already stored locally, or add resources generated by another workflow. Files prepared outside `hrtfpykit` can be placed under the dataset root and used together with official resources. This allows the same downloaded or user-provided files to be reused across different specifications without repeating the acquisition step.

Dataset construction is based on declarative specifications. Acoustic specifications define which HRTF-derived values should be returned, including time- or frequency-domain representations, selected positions, ears, frequency bins or bands, and binaural cues. They also define the indexing context of the sample, for example, whether each row corresponds to a subject, a subject-position pair, a subject-ear pair, or an individual time sample or frequency bin. Resource specifications align non-acoustic information with the same subjects, including anthropometric tables, metadata, meshes, images, and videos when those resources are available in the public dataset or supplied by the user. User-provided resources make dataset construction more flexible, allowing researchers to build multimodal samples for different HRTF-related problems; for example, an HRTF target can be paired with anthropometric measurements, ear images, or mesh resources from the same listener. The dataset builder scans only the required resource families, identifies the subjects for whom all required resources are available, applies exclusions, and creates deterministic training, validation, and test splits.

The resulting object behaves as a multimodal map-style dataset: its length is defined by the selected subjects and indexing context, and each indexed sample returns input values, target values, and metadata describing the dataset, subject, and active row context. This structure is designed to integrate naturally with common deep learning frameworks such as PyTorch [20], while remaining useful for inspection, preprocessing, and analysis outside a training loop. Acoustic values are loaded through the `hrtf` layer, so dataset samples use the same HRTF representations, transformations, selections, and metrics described above. Although HRTF individualization with deep learning is a central use case, the same mechanism also supports other HRTF-related learning and analysis tasks.

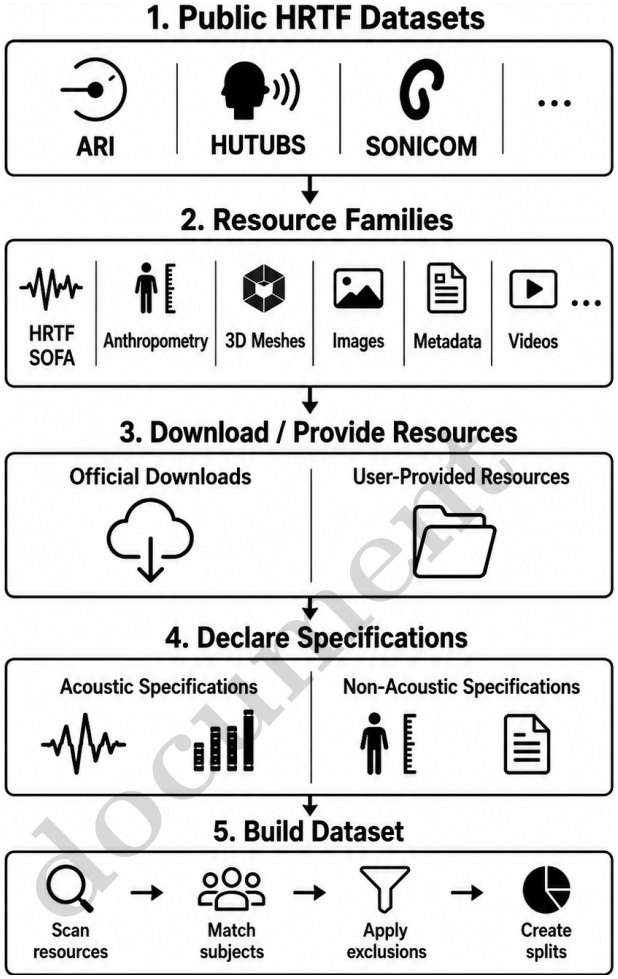

Figure 2: Logic of the `datasets` layer.

4. Representative workflows

The previous section described the API architecture of `hrtfpykit`. This section illustrates how the API layers can be combined in representative research tasks, from SOFA inspection and acoustic analysis to multimodal dataset construction for deep learning experiments.

4.1. HRTF analysis and visualization

The first workflow begins by loading a SOFA file as an HRTF object. The object retains access to the underlying SOFA representation while exposing synchronized impulse-response, transfer-function, and source-position data for acoustic analysis. From this object, a researcher can inspect metadata, select source directions, compute binaural cues or comparison metrics, apply transformations, and visualize the current state through the plotting layer. This workflow is useful for checking files from public datasets, validating preprocessing decisions, comparing alternative acoustic representations, and exporting processed HRTFs for later experiments. Listing 1 shows this workflow in a compact form. The example loads a SOFA file, inspects its metadata and acoustic representations, selects the horizontal plane, applies a time-domain

window, computes ITD values, creates a magnitude plot, and saves the processed HRTF to a new SOFA file.

```
from hrtfpykit.hrtf import itd, load_hrtf
from hrtfpykit.plots import plot_magnitude

hrtf = load_hrtf("example.sofa")
metadata = hrtf.Sofa.GlobalAttributes
ir_values = hrtf.IR.values
tf_values = hrtf.TF.values
source_positions = hrtf.Sources.get_positions()

horizontal = hrtf.select(plane="horizontal")
processed = horizontal.transform.apply_window("hann")
itd_values = itd(processed)
figure = plot_magnitude(processed, show=False)
processed.save("processed.sofa", overwrite=True)
```

Listing 1: HRTF workflow for loading, inspecting, transforming, plotting, and saving a SOFA file.

4.2. Dataset preparation for HRTF individualization

The second workflow uses the `datasets` layer to construct a training dataset for HRTF individualization. The researcher selects a public dataset, chooses the official resources to download, and declares input and target specifications. Listing 2 illustrates this idea with a SONICOM dataset that downloads HRTF SOFA files from the Imperial server and aligns them with user-provided ear images. The target is a frequency-domain HRTF magnitude representation, while the input image paths are resolved from an external image folder under the dataset root. The dataset builder matches subjects across the required resources, applies exclusions, creates the selected split, and returns map-style samples that can be inspected directly or passed to a deep learning framework.

```
from hrtfpykit.datasets import (
    HRTFSpec,
    ImageSpec,
    SONICOM,
)

dataset = SONICOM(
    root="sonicom",
    download=True,
    download_server="imperial",
    download_resources="hrtf",
    inputs=ImageSpec(
        path="ear_images",
        name="ear_image",
    ),
    target=HRTFSpec(
        domain="frequency",
        signal="tf_magnitude_db",
        index_by=("subject", "position"),
    ),
    split="train",
)
sample = dataset[0]
```

Listing 2: Dataset workflow for pairing user-provided ear images with downloaded HRTF targets.

5. Conclusion and future work

`hrtfpykit` brings together SOFA file handling, acoustic processing, visualization, access to public HRTF datasets, and data pipeline construction within a unified Python interface. By integrating these capabilities within a single library, it reduces the need for separate packages and scripts, creating a more consistent and reproducible workflow for HRTF analysis and deep learning experiments. Future work will focus on supporting additional public HRTF datasets and expanding the available acoustic transformations, metrics, and data-processing capabilities. We expect `hrtfpykit` to be useful to the spatial audio community, especially for research on HRTF individualization. Researchers and developers are welcome to contribute to the public codebase and help expand the project.

6. Acknowledgments

This work was supported by the Spanish Ministry of Science, Innovation and Universities under projects PID2022-137048OB-C42 and PID2025-174060OB-C32. We would like to thank everyone who provided feedback, support, and encouragement during the development of `hrtfpykit`.

References

[1] V. R. Algazi, R. O. Duda, R. Duraiswami, N. A. Gumerov, and Z. Tang, "Approximating the head-related transfer function using simple geometric models of the head and torso," *J. Acoust. Soc. Am.*, vol. 112, no. 5, pp. 2053–2064, 2002. DOI: 10.1121/1.1508780

[2] J. C. Middlebrooks, "Virtual localization improved by scaling nonindividualized external-ear transfer functions in frequency," *J. Acoust. Soc. Am.*, vol. 106, no. 3, pp. 1493–1510, 1999. DOI: 10.1121/1.427147

[3] H. Møller, M. F. Sørensen, C. B. Jensen, and D. Hammershøi, "Binaural technique: Do we need individual recordings?" *J. Audio Eng. Soc.*, vol. 44, no. 6, pp. 451–469, 1996.

[4] M. Mihocic and P. Majdak, *Sofa toolbox 2.4.0*, The SONICOM Ecosystem: Tool #14, 2025. DOI: 10.60887/q17dygeudms0aksh [Online]. Available: https://ecosystem.sonicom.eu/tools/14

[5] pyfar developers, *Sofar documentation*, 2026. [Online]. Available: https://sofar.readthedocs.io/

[6] P. Majdak et al., "Spatially oriented format for acoustics: A data exchange format representing head-related transfer functions," in *134th Audio Engineering Society Convention*, Rome, Italy, 2013.

[7] J. Pauwels, "The hartufo toolkit for machine learning with hrtf data," in *Proc. AES International Conference on Spatial and Immersive Audio*, Huddersfield, UK, 2023.

[8] K. C. Poole et al., *The extended sonicom hrtf dataset and spatial audio metrics toolbox*, 2025. arXiv: 2507.05053. [Online]. Available: https://doi.org/10.48550/arXiv.2507.05053

[9] G. W. Lee and H. K. Kim, "Personalized hrtf modeling based on deep neural network using anthropometric measurements and images of the ear," *Appl. Sci.*, vol. 8, no. 11, p. 2180, 2018. DOI: 10.3390/app8112180

[10] Y. Wang, Y. Zhang, Z. Duan, and M. Bocko, "Global hrtf personalization using anthropometric measures," in *150th Audio Engineering Society Convention*, 2021.

[11] J. Zhao, D. Yao, J. Gu, and J. Li, "Efficient prediction of individual head-related transfer functions based on 3d meshes," *Appl. Acoust.*, vol. 219, p. 109 938, 2024. DOI: 10.1016/j.apacoust.2024.109938

[12] A. Alvarez-Martinez and J. J. Lopez, "Hrtf individualization using ear images from 3d meshes," in *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, vol. 270, 2024. DOI: 10.3397/IN_2024_3741

[13] A. Alvarez-Martinez, "Towards practical hrtf individualization: Machine learning approaches using visual data," Ph.D. dissertation, Universitat Politecnica de Valencia, Dec. 3, 2025. DOI: 10.4995/thesis/10251/231530

[14] P. Majdak, F. Zotter, F. Brinkmann, J. D. Muynke, M. Mihocic, and M. Noisternig, "Spatially oriented format for acoustics 2.1: Introduction and recent advances," *J. Audio Eng. Soc.*, vol. 70, no. 7/8, pp. 565–584, 2022. DOI: 10.17743/jaes.2022.0026

[15] J. Blauert, *Spatial Hearing: The Psychophysics of Human Sound Localization*, Revised edition. Cambridge, MA, USA: MIT Press, 1996. DOI: 10.7551/mitpress/6391.001.0001

[16] J. C. Middlebrooks, "Individual differences in external-ear transfer functions reduced by scaling in frequency," *J. Acoust. Soc. Am.*, vol. 106, no. 3, pp. 1480–1492, 1999. DOI: 10.1121/1.427176

[17] P. Majdak, M. J. Goupell, and B. Laback, "3-d localization of virtual sound sources: Effects of visual environment, pointing method, and training," *Atten. Percept. Psychophys.*, vol. 72, no. 2, pp. 454–469, 2010. DOI: 10.3758/APP.72.2.454

[18] F. Brinkmann et al., "A cross-evaluated database of measured and simulated hrtfs including 3d head meshes, anthropometric features, and headphone impulse responses," *J. Audio Eng. Soc.*, vol. 67, no. 9, pp. 705–718, 2019.

[19] I. Engel et al., "The sonicom hrtf dataset," *J. Audio Eng. Soc.*, vol. 71, no. 5, pp. 241–253, 2023. DOI: 10.17743/jaes.2022.0066

[20] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html