SoftwareX 31 (2025) 102306

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

MetrikaBox: An open framework for experimenting with audio classification

Jorge Perianez-Pascual [a,*], Juan D. Gutiérrez [b], Emilio Delgado [a], Fernando Sánchez-Figueroa [a], Roberto Rodriguez-Echeverria [a]

[a] Instituto de Investigación en Tecnologías Informáticas Aplicadas (INTIA), Universidad de Extremadura, Av. Universidad s/n, 10003 Cáceres, Spain
[b] Department of Electronics and Computer Science, Universidade de Santiago de Compostela, Rúa Benigno Ledo, 27002, Lugo, Spain

ARTICLE INFO

Keywords:
Artificial Intelligence
Deep Learning
Audio classification
Neural networks
Digital signal processing
Software engineering

ABSTRACT

This paper presents MetrikaBox, a general-purpose, open-source, and extensible audio classification package designed to facilitate the development of Deep Learning (DL) models for a wide range of audio processing tasks. The software manages all necessary preprocessing steps to build classification models capable of distinguishing between user-defined classes using advanced Artificial Intelligence (AI) techniques. MetrikaBox is well suited for tasks such as musical genre classification, voice-versus-music discrimination, and other audio classification or segmentation applications. Users can either employ the package as provided or extend it by integrating their own datasets, classification models, data loading systems, augmentation techniques, and more. The package has been tested in both commercial and academic settings, where it has produced models for industrial audio processing and served as a platform for proof-of-concept applications. Comprehensive documentation and practical examples included in the repository support users in integrating the system into their audio analysis projects. MetrikaBox is openly available and provides a user interface for convenient testing.

Code metadata

Current code version	1.1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-25-00308
Permanent link to Reproducible Capsule	–
Legal Code License	GNU General Public License (GPL)
Code versioning system used	git
Software code languages, tools, and services used	python
Compilation requirements, operating environments & dependencies	https://github.com/i3uex/metrikabox/blob/main/requirements.txt
If available Link to developer documentation/manual	https://github.com/i3uex/metrikabox/blob/main/README.md
Support email for questions	jpery@unex.es

1. Motivation and significance

In recent years, the content we consume for both educational and entertainment purposes has undergone a radical transformation, becoming predominantly multimedia. Within this context, audio plays a pivotal role—whether as music, podcasts, audiobooks, or as an integral component of audiovisual media. The extraction of features from audio signals has been a topic of sustained interest in the scientific community for decades. These features capture characteristics such as the timbral texture of a sound or its frequency distribution. Such representations enable a variety of downstream tasks, including speaker identification, emotion recognition, and, ultimately, audio classification, wherein sounds are categorized into predefined classes based on their extracted attributes.

Audio classification has become a commonplace task in many aspects of everyday life. It encompasses a wide range of applications, from distinguishing between speech and music to the automatic generation of subtitles, musical genre recognition, and enabling virtual assistants such as Alexa or Siri to interpret spoken commands. The underlying process involves assigning a label to an audio fragment based on its extracted features. This classification task can range from simple distinctions, such as separating speech from music, to more

* Corresponding author.
E-mail address: jpery@unex.es (J. Perianez-Pascual).

https://doi.org/10.1016/j.softx.2025.102306
Received 16 May 2025; Received in revised form 1 August 2025; Accepted 8 August 2025
Available online 28 August 2025

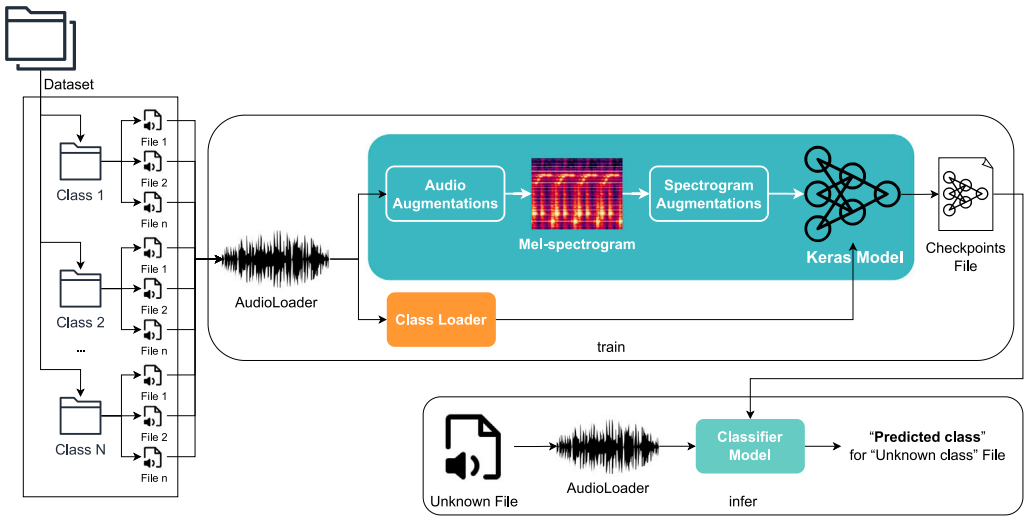

Fig. 1. Overview of the MetrikaBox architecture.

complex analyses, including genre classification, style identification, tempo detection, voice presence, or even mood inference.

Traditionally, audio classification was approached using probabilistic methods that relied on manual feature extraction followed by statistical analysis. Tools such as pyAudioAnalysis [1], Essentia [2], and librosa [3] provide established solutions for this feature extraction process. However, the advent of Artificial Intelligence (AI) has transformed the field, enabling advanced techniques that perform automatic classification without the need for explicit feature extraction.

With the widespread adoption of Convolutional Neural Networks (CNNs) for image classification tasks [4], it has become feasible to treat audio signals as images through their spectral representations [5]. This approach allows an audio file to be divided into segments, which are then converted into spectrograms and subsequently classified using neural networks. As a result, this methodology supports both audio classification and segmentation.

Developing Deep Learning (DL) models for audio classification without the aid of specialized tools is a highly complex endeavor. It involves, among other steps, data preprocessing (including normalization, segmentation, and, in some cases, feature extraction), the design of suitable neural network architectures, meticulous configuration of the training pipeline (*e.g.* , optimization strategies and regularization techniques), rigorous evaluation using appropriate metrics, and preparation of the model for deployment. This workflow requires advanced expertise across multiple domains such as audio engineering, programming, and DL, posing a significant barrier for non-specialist users.

Despite the availability of various libraries, a gap remains for a tool that integrates the entire workflow in an accessible environment. For instance, powerful proprietary platforms like MATLAB offer extensive toolboxes for DL and audio processing,[1] including specialized extensions for tasks like data augmentation [6]. However, these are mostly collections of functions that require significant programming expertise to build a complete classification pipeline. Similarly, while open-source libraries like pyAudioAnalysis focus on classical machine learning, and comprehensive frameworks like TensorFlow [7], Keras [8], and PyTorch [9] provide the necessary components, they are intended for users with strong programming and architecture design skills. None of these alternatives provides a unified, user-friendly environment that automates the entire process from data loading to inference.

In this context, we propose MetrikaBox, a tool that leverages DL to integrate and automate all stages of the audio classification process, including data loading, model training, and final model export for inference, thereby reducing technical complexity and improving usability. This solution is distinguished by its no-code approach, offering predefined neural network architectures and an intuitive interface that simplifies the setup and execution of experiments. Users can choose between a programmatic Application Programming Interface (API), a Command-Line Interface (CLI), and Gradio [10] graphical interfaces to use MetrikaBox. It is specifically designed to enable rapid experimentation by non-specialist users without compromising the quality or performance of the resulting models. A visual overview of the system is presented in Fig. 1.

The remainder of this paper is organized as follows. Section 2 describes the architecture of MetrikaBox. An illustrative example demonstrating its practical use with generic example datasets is presented in Section 3. The impact and benefits of using MetrikaBox are discussed in Section 4. Finally, Section 5 concludes the paper and outlines directions for future work.

2. Software description

MetrikaBox is a Python-based toolkit designed to train, deploy, and apply DL models for audio classification and segmentation. Built on top of Keras, it offers a modular and extensible architecture that allows users to adapt and customize every part of the pipeline — from data loading and preprocessing to model design and inference — making it suitable for both research and production environments.

The software supports a wide range of audio formats, including raw audio files and compressed formats such as EnCodec [11], and is designed to be flexible enough to handle a variety of tasks, from speech/music discrimination to fine-grained segmentation. MetrikaBox is built in Python 3.10 and is compatible with Windows, Linux, and macOS environments.

2.1. Software architecture

MetrikaBox is structured around independent modules, as illustrated in Fig. 2.

Each module is responsible for a key part of the pipeline. The Data Augmentation module (`augmentations`) contains custom Keras-compatible layers for dynamic augmentation of audio data during training, helping improve model generalization by simulating diverse audio conditions. The Model Management module (`model`) includes

[1] https://mathworks.com/help/deeplearning/audio-processing.html

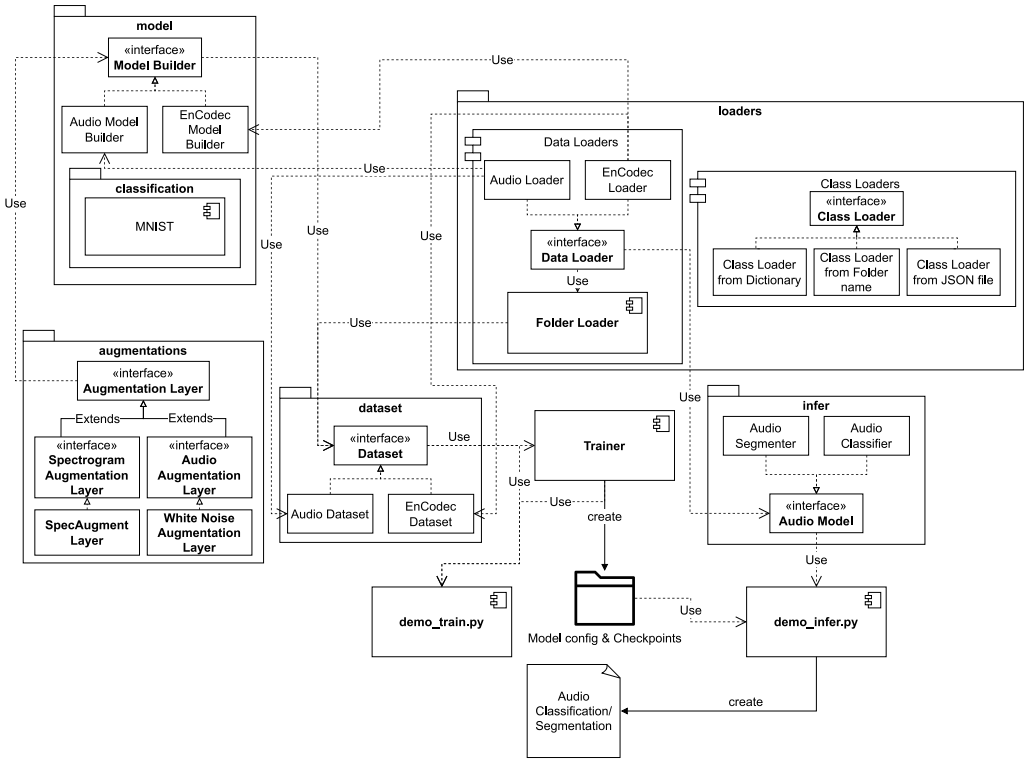

Fig. 2. Overview of MetrikaBox software modules.

builders for handling different audio formats (*e.g.* , raw audio or EnCodec) and adapts them for training. It also provides a classification submodule offering a simple example model (based on the Simple MNIST Convnet). A key feature of MetrikaBox is its direct support for integrating and fine-tuning pre-trained models from Keras Applications.[2] The selection of the model architecture, such as MobileNetV3 or ResNet50, is a primary parameter in the user interface, enabling users to leverage state-of-the-art architectures to enhance performance and reduce training time. In addition, the framework's extensible design allows advanced users to integrate their own custom models and neural network architectures, tailoring the pipeline to specific research needs. The Data Loading module (`loaders`) is composed of two subpackages: on one hand, Class Loaders, which assign labels based on folder names, an external mapping, or a JavaScript Object Notation (JSON) file alongside each audio file; on the other hand, Data Loaders, which manage audio file loading and support both raw formats (MP3, WAV, OGG) and compressed files (EnCodec format). The Dataset Management module (`dataset`) provides abstractions for managing different types of datasets: `AudioDataset` for raw audio files (*e.g.* , MP3, WAV, OGG) and `EnCodecDataset` for compressed audio content. The Inference module (`infer`) handles the inference process, with the `AudioModel` class at its core. This class loads audio data, applies the trained model, and supports both segmentation (splitting audio into small labeled sections) and full-file classification (assigning a single label to the entire file).

The software is structured around demo scripts. Training and inference tasks can be performed via the CLI for scripting or integration purposes using `main.py`. The Gradio graphical interfaces for training and inference can be launched from `demo_train.py` and `demo_infer.py`, respectively.

This modular design ensures that each component can be extended or replaced easily, allowing advanced users to integrate new models, add support for additional audio formats, or customize the inference process as needed.

2.2. Available interfaces

MetrikaBox is designed with a triple-interface system to meet the needs of different user profiles. First, the programmatic API: all core functionalities can be accessed directly through Python modules. This makes MetrikaBox highly extensible and suitable for seamless integration into custom projects, research experiments, or larger machine learning pipelines. Second, the CLI: the `main.py` script provides a straightforward way to run both training and inference tasks from the terminal. This interface is ideal for scripting, automation, and integration into workflows where graphical interaction is not practical. Alternatively, two Gradio graphical interfaces (`train.py` and `infer.py`) allow users to configure training and inference tasks interactively via a web browser. These interfaces are designed to lower the barrier to entry, enabling non-programmers to experiment with models and visualize results without writing code.

2.3. Software functionality

MetrikaBox provides full control over the training process, allowing users to configure audio preprocessing settings (*e.g.* , sample rate, window size, step size), training parameters (*e.g.* , batch size, number of epochs), model architecture, data augmentation strategies, and spectrogram transformation options.

Once training is complete, the resulting Keras model and its corresponding configuration file are made available. Within the Gradio web interface, these files can be downloaded directly and subsequently uploaded for use in the inference interface. For more advanced applications and integration into custom workflows, MetrikaBox also provides a programmatic API. This allows the models to be saved to user-specified paths and loaded for deployment, offering the flexibility to

[2] https://keras.io/api/applications/

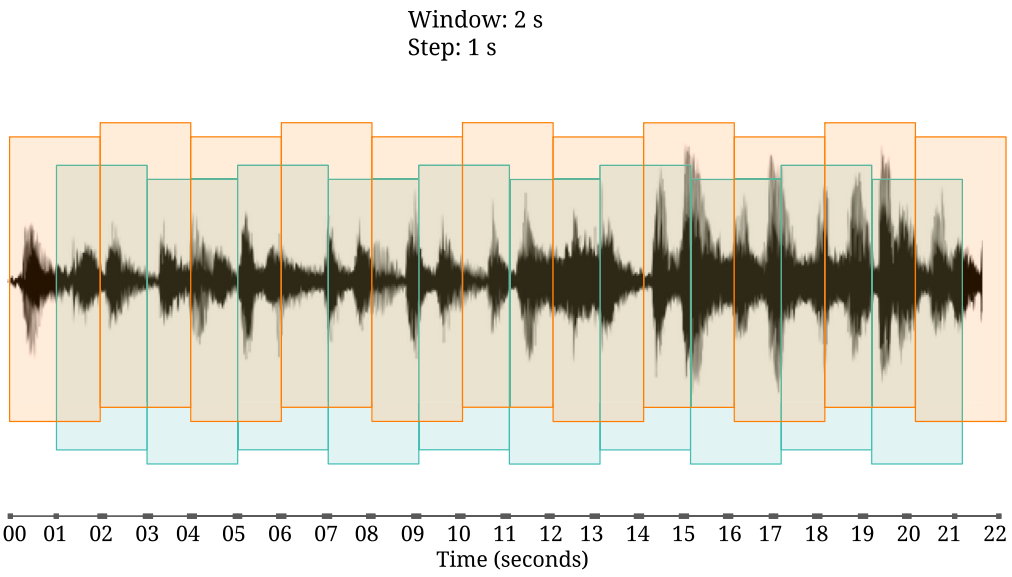

Fig. 3. Illustration of the sliding window process.

incorporate the trained classifiers into a variety of external applications. In inference mode, users can classify entire audio files or segment them into labeled sections. Results can be presented interactively in the Gradio interface (for example, as lists of predicted classes with their probabilities), saved in JSON files (for segmentation), or accessed programmatically for further processing.

A primary consideration for deployment is that external applications must replicate the same configuration used within MetrikaBox, such as sample rate, window size, and step size, to ensure model compatibility and performance. To facilitate this, users can leverage the JSON configuration file that MetrikaBox provides upon training completion, which details all necessary parameters for reproducibility. Furthermore, the deployment environment must have the necessary software dependencies, such as Keras and TensorFlow, installed.

2.4. Audio preprocessing

Audio preprocessing is a critical component of MetrikaBox, transforming raw audio files into model-ready inputs. The framework grants users detailed control over crucial parameters to adapt the analysis for specific tasks. Key settings such as the target sample frequency, the time window for spectral analysis, and the step size are fully configurable. This allows for experimentation based on the specific requirements of the audio task. These parameters can be configured in the different versions of the software before initiating training. This configurability is essential for handling the distinct requirements of different audio types, such as voice and music. While the default spectral representation is the mel spectrogram, chosen for its perceptual relevance, the framework's modular design is intentionally extensible. MetrikaBox also supports an alternative pipeline using latent space representations from the EnCodec neural audio codec. Furthermore, the architecture allows advanced users to integrate other representations such as standard spectrograms, Log-Mel spectrograms, or Mel-Frequency Cepstral Coefficientss (MFCCs) to suit their experimental needs.

When a file is loaded, it is first converted to mono (if multichannel) and then segmented using a sliding window approach. This process splits the audio into overlapping segments, with the window and step sizes configurable by the user, ensuring both detailed coverage and retention of temporal context. For example, with a 5-second window and a 2.5-second step (illustrated in Fig. 3), each segment overlaps the next by 50 %, allowing the model to capture gradual transitions in the audio. Padding is applied as needed to ensure that all segments maintain a consistent length, which is essential for uniform model input.

This preprocessing method enables two key functionalities: segmentation tasks, where each segment is independently classified, and full-file classification, where predictions across segments are aggregated to determine an overall label for the audio file.

2.5. Neural network architecture

MetrikaBox provides a default DL pipeline based on Keras (Fig. 4), which users can customize or extend as needed. The architecture processes batches of audio segments, beginning with normalization to `float32` format scaled to $[-1, 1]$. Data augmentation layers can be applied both before and after the transformation of audio into spectrogram form.

Since MetrikaBox is built on top of Keras, it automatically detects and utilizes the most suitable computation device available, whether it be a GPU, CPU, or other compatible hardware. This process is handled seamlessly by the underlying framework, requiring no specific configuration from the user to leverage the optimal hardware for accelerating model training.

A core component of the pipeline is the mel spectrogram transformation, which converts audio segments into 2D spectrogram images. This was originally implemented using the Kapre library [12]; however, with the release of Keras 3.1, MetrikaBox now utilizes the native mel spectrogram layer for tighter integration and improved performance. After transformation, spectrograms pass through additional augmentation layers if specified.

In addition to the spectrogram-based approach, MetrikaBox also supports a latent space pipeline using EnCodec [13], a neural audio codec. In this alternative workflow, the compressed latent representations generated by EnCodec are fed directly into the DL model, bypassing traditional spectrogram conversion. This method leverages the compact, information-rich nature of EnCodec's latent space, offering a promising alternative for efficient audio classification with potentially reduced preprocessing overhead. Thanks to the toolkit's extensible architecture, users can implement and integrate other audio representations of their choice, enabling experimentation with novel feature spaces or adaptation of the pipeline to specialized use cases.

For both approaches, a `Conv2D` layer adapts the input to a 3-channel format, ensuring compatibility with image-based classification

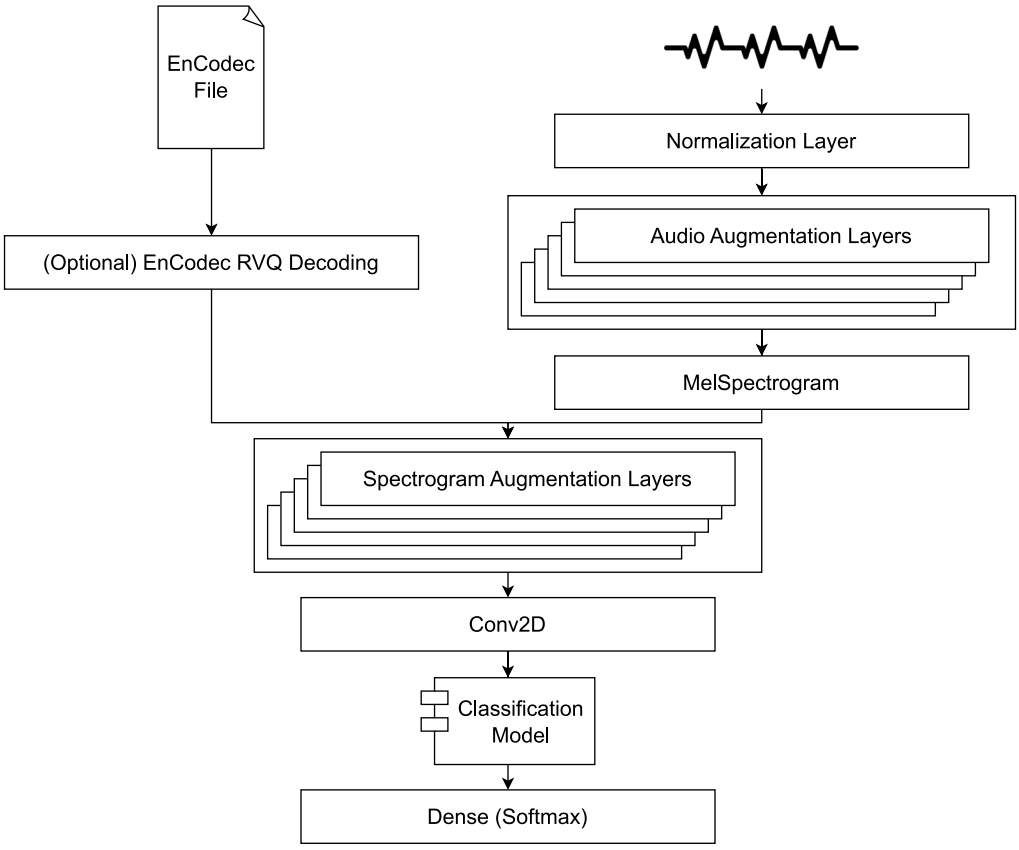

Fig. 4. Neural network architecture diagram.

models. The processed data is then fed into the user-selected model (which can be a custom or pre-trained architecture), and a final `Dense` layer with softmax activation produces the classification output.

In segmentation mode, each segment is processed individually to create a fine-grained labeled timeline of the audio. In full-file classification mode, the system averages predictions across all segments to provide a robust overall classification result.

3. Illustrative examples

To demonstrate MetrikaBox's capabilities, we present examples of both segmentation and full-file classification tasks.

When using Gradio for the training phase, the interface guides users through the complete process of model development. Without writing a single line of code, users can select an audio dataset, choose a neural network architecture, configure hyperparameters such as batch size and number of epochs, and launch training directly from the browser. Once training is complete, the interface displays a comprehensive set of performance metrics to help users evaluate the model's quality. Beyond accuracy and loss, the tool also computes and visualizes precision and recall, as shown in Fig. 5, and supports other standard metrics like the F1-score. Furthermore, thanks to its modular design built on Keras, users can easily incorporate any other existing metric available in the Keras library or even program their own for highly customized evaluations. This feedback enables users to make more informed adjustments in future iterations. In Fig. 5, a MobileNetV3Small model is being trained using the GTZAN Genres dataset through the Gradio training interface. To initiate training, the user specifies the dataset path, which must point to a directory containing one subfolder per class (*e.g.* , `speech/`, `music/`), each populated with the corresponding audio files. These folder names are automatically interpreted as class labels during model training. After selecting the model architecture, the user can configure additional training parameters such as batch size, number of epochs, and learning rate, as well as audio processing parameters like window size and step size, which determine how the audio is split into overlapping frames for analysis. In this example, a 5-second window with a 2.5-second step is used to provide sufficient context for classification while maintaining dense coverage of the input signal. Once initiated, the model is trained on the dataset without requiring manual scripting or command-line input. This interface offers an accessible way to experiment with different datasets and model configurations, making it particularly suitable for researchers, educators, and practitioners seeking rapid prototyping or reproducible experimentation in audio classification tasks.

The same training process can also be executed using the software's CLI, offering full control for users who prefer scriptable workflows or require automation. This mode is particularly useful for integrating training into larger pipelines, running batch experiments, or deploying models in headless environments where a graphical interface is not practical.

The inference Gradio interface is designed for evaluating previously trained models on new audio content. Users can upload or specify an audio file, select a pre-trained model, and choose between segmenting the file or performing global classification. Once the analysis is complete, results are presented in a visual, interactive format that includes timeline segmentations, class summaries, and prediction confidence scores. This interface is particularly useful for users seeking quick insights or comparative model evaluation without technical overhead.

In Fig. 6, the Gradio inference interface is shown in action, showcasing the classification of an audio file using the selected pre-trained model. In this example, the user begins by specifying the path to the audio file or uploading it directly, and then uploads the pre-trained model along with its configuration file, which must be downloaded after the training process via the training interface. After launching the

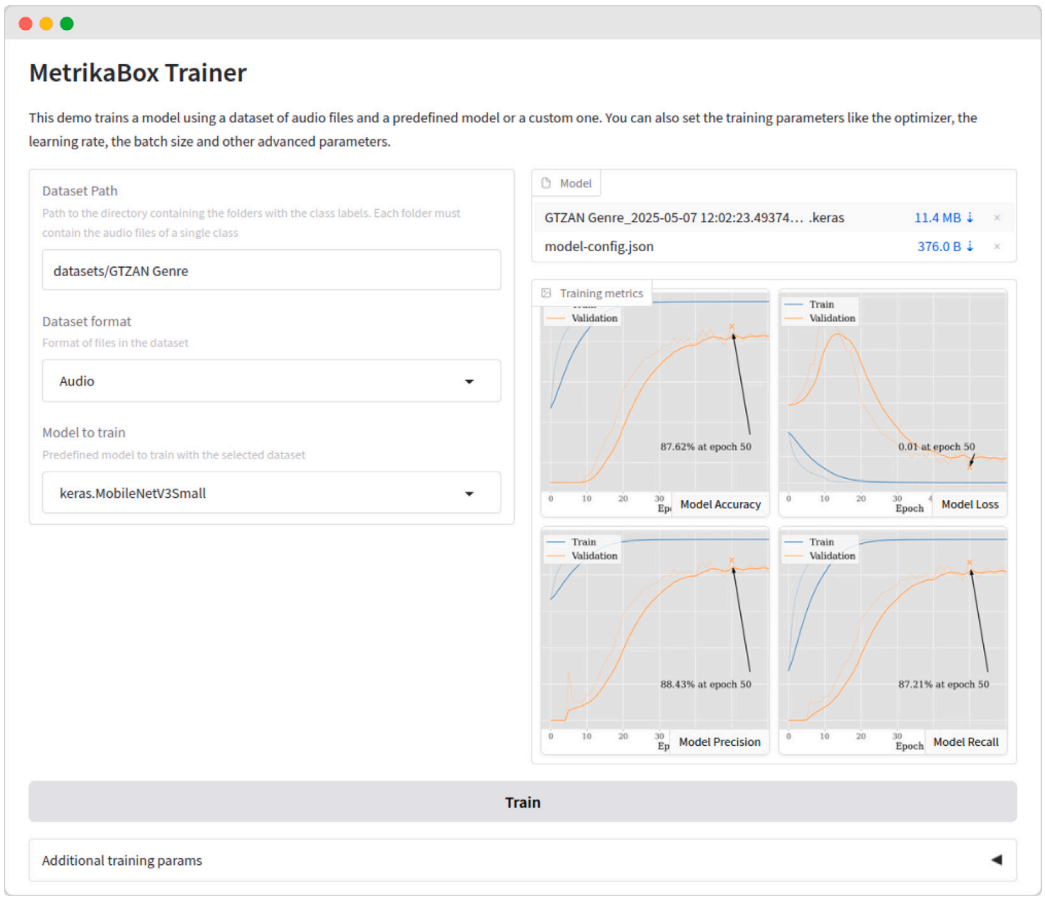

Fig. 5. Screenshot of the Gradio interface for training.

analysis, the system processes the file and presents the results in an intuitive format. For full-file classification, it displays a list of predicted classes along with their associated probabilities, providing a clear summary of the model's output. When performing segmentation tasks, the system generates a JSON file as output, detailing each detected segment along with its corresponding predicted class.

The same inference process can also be performed using the software's CLI, providing a flexible option for users who prefer scripting or need to integrate audio classification into automated workflows.

To assess its performance in controlled settings, the software was tested on classifying musical genres from short audio clips. Despite the limited duration of the samples and stylistic overlap between genres, the model consistently achieved validation accuracy above 85 %, successfully identifying genres such as classical, jazz, and metal. As shown in Fig. 7, the training process converged reliably, supporting the model's effectiveness in broader music information retrieval tasks.

The reported validation accuracy of 87.62 % on the GTZAN Genres dataset was achieved using the MobileNetV3Small model, a lightweight and efficient architecture suitable for rapid experimentation. This result is highly competitive and compares favorably with state-of-the-art benchmarks on the same dataset, such as the 86.30 % accuracy reported by Kong et al. [14] and the 86.10 % by Niizumi et al. [15]. A more detailed comparative analysis can be found in our related work [13].

It is important to note that as a flexible framework, MetrikaBox allows users to select any Keras-compatible model. Therefore, key metrics such as model size (number of parameters) and training time are not fixed; they are inherently dependent on the user's choice of model architecture, the size of the dataset, and the computational hardware available (*e.g.* , CPU or GPU). The framework is designed precisely to empower users to make these choices based on their specific performance requirements and resource constraints.

Beyond controlled evaluations, the software has also been deployed in real-world broadcasting settings. In commercial radio stations, it processes 24 h of audio content per station, automatically segmenting streams into speech, music, and mixed content. It distinguishes between foreground and background music usage and enables track identification from detected music segments. The structured reports generated by the system support compliance with national copyright regulations and contribute to a more transparent, data-driven approach to royalty distribution. This automation has significantly reduced manual annotation workloads while delivering consistent and auditable results. The system operates reliably in both live and pre-recorded environments, adapting seamlessly to programming that includes talk shows, music blocks, news, and advertising.

4. Impact

MetrikaBox's development has been supported by the European Union through the ADAPIMMA and MusicGenia projects, which together have provided 289 921.57 e in funding. ADAPIMMA is a regionally funded project focused on developing techniques to monitor the precise use of music in radio broadcasts, ensuring that royalties are distributed fairly among authors in accordance with Spain's updated Intellectual Property Law. MusicGenia, supported by European funding, aims to develop a cloud-based platform for AI-generated production music, offering content creators access to royalty-free, original music tailored to their needs. The MetrikaBox framework is employed in both projects.

One of the most tangible outcomes of MetrikaBox has been the creation of MetrikaMedia,[3] a spin-off company from the Universidad

[3] https://metrika.media

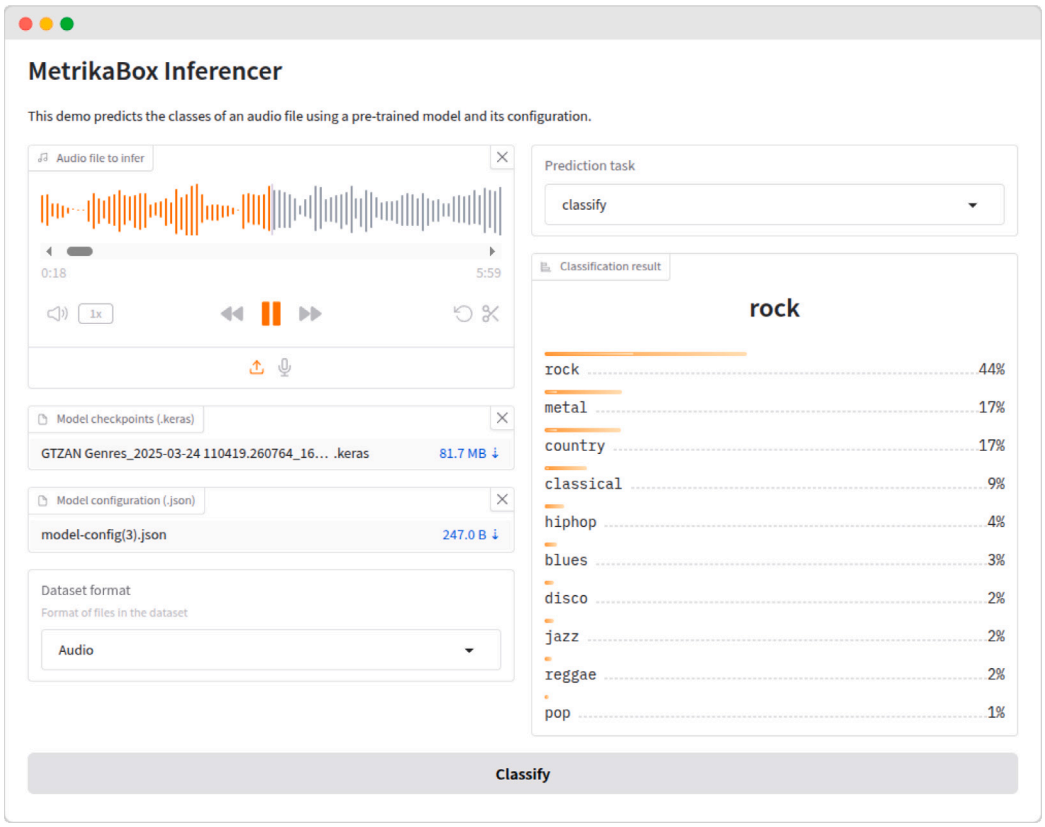

Fig. 6. Screenshot of the Gradio inference interface analyzing 'Bohemian Rhapsody' by Queen.

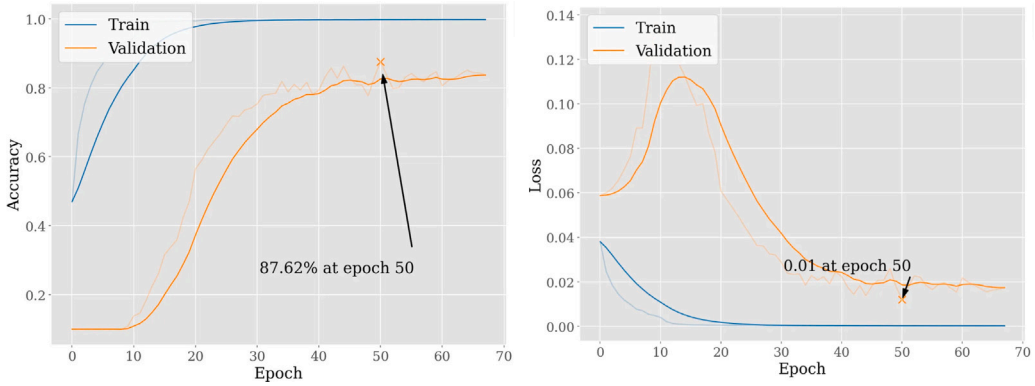

Fig. 7. Accuracy and loss metrics for the GTZAN Genres dataset.

de Extremadura founded in 2020. The MetrikaBox software framework is extensively used within the company. MetrikaMedia has successfully deployed the software in major radio stations across Spain and Latin America, where it has become an essential tool for automating the monitoring and analysis of audio content. With its ability to perform audio classification, identify specific tracks, and assess the prominence of background versus foreground music, the software enables broadcasters to comply with regulatory requirements while contributing to a more transparent and equitable royalty distribution system for artists.

Importantly, the data collected and processed by this software has also fueled broader research innovations. For example, it enabled the development of a hybrid recommender system for radio programs [16], which personalizes content recommendations by combining user preferences with historical listening data. This work demonstrated how audio analytics can be harnessed not only for compliance but also to enhance the listener experience. Additionally, this research led to the publication of a second paper, which introduced a novel approach to audio classification using latent representations from Meta's EnCodec neural audio codec [13]. This study showed that such representations can achieve competitive performance with significantly improved efficiency compared to traditional spectrogram-based approaches. Together, these publications highlight the scientific versatility and impact of the technologies developed.

Finally, the open-source release of MetrikaBox amplifies the software's reach within the research community. By providing accessible tools for audio preprocessing and classification, it lowers technical barriers and enables researchers to develop new models, conduct reproducible experiments, and explore applications ranging from music information retrieval to sound event detection.

5. Conclusions

This paper introduced MetrikaBox, an AI-based toolkit for automated audio content analysis. Designed to support both applied use cases and research workflows, MetrikaBox combines DL-based classification and segmentation with a flexible, extensible architecture. It enables users to analyze, label, and manage large volumes of audio data — whether for compliance, content organization, or machine learning experiments — without requiring deep technical expertise.

The toolkit provides three modes of interaction: intuitive Gradio-based web interfaces for training and inference, a command-line interface (CLI) for automation and scripting, and direct Python-level access for seamless integration into custom codebases and experimental pipelines. This versatility makes it accessible to a wide range of users, from media professionals to data scientists. MetrikaBox has been validated on benchmark datasets, demonstrating robust performance in tasks such as speech/music discrimination and genre classification, and has proven its reliability in continuous-use, real-time processing environments.

Future development will focus on extending task coverage — including speaker identification, emotion recognition, and multilingual classification — while improving usability through guided workflows and pre-configured models. Real-time streaming support and enhanced model interpretability are also key goals. We hope this work enables a new wave of open, accessible, and intelligent audio analysis tools that foster innovation across research, industry, and creative domains.

CRediT authorship contribution statement

Jorge Perianez-Pascual: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Juan D. Gutiérrez:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Emilio Delgado:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Fernando Sánchez-Figueroa:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Roberto Rodriguez-Echeverria:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Roberto Rodriguez-Echeverria reports financial support was provided by Spain Ministry of Science and Innovation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by Grant CPP2021-008491 funded by MICIU/AEI/10.13039/50100011033 and by the European Union NextGeneration EU/PRTR.

References

[1] Giannakopoulos T. pyAudioAnalysis: An open-source python library for audio signal analysis. PLoS One 2015;10(12):e0144610. http://dx.doi.org/10.1371/journal.pone.0144610.

[2] Bogdanov D, Wack N, Gomez E, Gulati S, Herrera P, Mayor O, et al. Essentia: An audio analysis library for music information retrieval. In: 14th conference of the international society for music information retrieval. 2013, p. 493–8. http://dx.doi.org/10.1145/2502081.2502229, arXiv:10230/32252.

[3] McFee B, Raffel C, Liang D, Ellis D, McVicar M, Battenberg E, et al. Librosa: Audio and music signal analysis in python. In: Python in science conference. 2015, p. 18–24. http://dx.doi.org/10.25080/Majora-7b98e3ed-003.

[4] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM 2017;60(6):84–90. http://dx.doi.org/10.1145/3065386.

[5] Zhang B, Leitner J, Thornton S. Audio recognition using mel spectrograms and convolution neural networks. San Diego, CA, USA: Noiselab University of California; 2019, p. 5.

[6] Maguolo G, Paci M, Nanni L, Bonan L. Audiogmenter: A MATLAB toolbox for audio data augmentation. Appl Comput Inform 2025;21(1/2):152–63. http://dx.doi.org/10.1108/ACI-03-2021-0064.

[7] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. TensorFlow: A system for large-scale machine learning. 2016, http://dx.doi.org/10.48550/arXiv.1605.08695, arXiv:1605.08695.

[8] Chollet F, et al. Keras. 2015.

[9] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. 2019, http://dx.doi.org/10.48550/arXiv.1912.01703, arXiv:1912.01703.

[10] Abid A, Abdalla A, Abid A, Khan D, Alfozan A, Zou J. Gradio: hassle-free sharing and testing of ML models in the wild. 2019, http://dx.doi.org/10.48550/arXiv.1906.02569, arXiv:1906.02569.

[11] Défossez A, Copet J, Synnaeve G, Adi Y. High fidelity neural audio compression. 2022, http://dx.doi.org/10.48550/arXiv.2210.13438, arXiv:2210.13438.

[12] Choi K, Joo D, Kim J. Kapre: on-GPU audio preprocessing layers for a quick implementation of deep neural network models with keras. 2017, http://dx.doi.org/10.48550/arXiv.1706.05781, arXiv:1706.05781.

[13] Perianez-Pascual J, Gutiérrez JD, Escobar-Encinas L, Rubio-Largo Á, Rodriguez-Echeverria R. Beyond spectrograms: rethinking audio classification from EnCodec's latent space. Algorithms 2025;18(2). http://dx.doi.org/10.3390/a18020108.

[14] Kong Q, Cao Y, Iqbal T, Wang Y, Wang W, Plumbley MD. PANNs: Large-scale pretrained audio neural networks for audio pattern recognition. IEEE/ACM Trans Audio Speech Lang Process 2020;28:2880–94. http://dx.doi.org/10.1109/TASLP.2020.3030497.

[15] Niizumi D, Takeuchi D, Ohishi Y, Harada N, Kashino K. Masked spectrogram modeling using masked autoencoders for learning general-purpose audio representation. 2022, http://dx.doi.org/10.48550/arXiv.2204.12260, arXiv:2204.12260.

[16] Fernández-García AJ, Rodriguez-Echeverria R, Preciado JC, Perianez J, Gutiérrez JD. A hybrid multidimensional Recommender System for radio programs. Expert Syst Appl 2022;198:116706. http://dx.doi.org/10.1016/j.eswa.2022.116706.