Check for updates

Received: 15 April 2026 | Accepted: 7 August 2026

DOI: 10.1111/2041-210x.70406

Methods in Ecology and Evolution

APPLICATION

Bacpipe: A Python package to make bioacoustic deep learning models accessible

Vincent S. Kather[1,2,3] | Sylvain Haupert[4] | Burooj Ghani[1] | Dan Stowell[1,2,3]

[1]Naturalis Biodiversity Center, Leiden, The Netherlands

[2]Department of Intelligent Systems, Tilburg University, Tilburg, The Netherlands

[3]Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands

[4]Muséum Nationale d'Histoire Naturelle, Paris, France

Correspondence
Vincent S. Kather
Email: vincent.kather@naturalis.nl

Funding information
H2020 Marie Skłodowska-Curie Actions, Grant/Award Number: 101116715

Handling Editor: Laura Jane Graham

Abstract

1. Natural sounds have been recorded for millions of hours over the previous decades using passive acoustic monitoring. Improvements in deep learning models have vastly accelerated the analysis of large portions of this data. While new models advance the state-of-the-art, accessing them using tools to harness their full potential is not always straightforward.
2. Here we present *bacpipe*, a collection of bioacoustic deep learning models and evaluation pipelines accessible through a graphical and programming interface, designed for both ecologists and computer scientists.
3. *Bacpipe* streamlines the usage of state-of-the-art models on custom audio datasets, generating acoustic feature vectors (embeddings) and classifier predictions. A modular design allows evaluation and benchmarking of models through interactive visualizations, clustering and probing.
4. We believe that access to new deep learning models is important. By designing *bacpipe* to target a wide audience, researchers will be enabled to answer new ecological and evolutionary questions in bioacoustics.

KEYWORDS
bioacoustics, biodiversity monitoring, clustering, deep learning, ecoacoustics, embeddings, PAM

1 | INTRODUCTION

In bioacoustics, recordings of natural soundscapes span massive spatial and temporal scales (Wall et al., 2021), the analysis of which has for a long time relied on automated processing for animal species detection (Baumgartner & Mussoline, 2011) and soundscape characterization (Sueur et al., 2014).[1] Bioacoustic recordings are evaluated to study evolutionary and ecological phenomena, that is animal behaviour (Teixeira et al., 2019), interaction with environments (Desjonquères et al., 2024; Sebastián-González & Pérez-Granados, 2025), sound production (Seyfarth & Cheney, 2010) and more. Data are commonly gathered using passive acoustic monitoring (PAM) setups (Calupca et al., 2000; Mellinger et al., 2007; Sugai et al., 2019). Studying changes of phenomena in large datasets was previously restricted to portions of datasets that were annotated manually by bioacousticians listening to

[1]https://github.com/bioacoustic-ai/bioacoustics-datasets.

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.
© 2026 The Author(s). *Methods in Ecology and Evolution* published by John Wiley & Sons Ltd on behalf of British Ecological Society.

recordings (Sugai et al., 2019). This precise study of recordings is a time-consuming practice, that therefore limits the amount of data that can be processed (Mellinger et al., 2007; Sugai et al., 2019). In recent years, the rapid advance of deep learning in terrestrial, freshwater and marine bioacoustics has opened up the opportunity to develop tools to process and organize large acoustic datasets which has vastly improved automated species detection (Stowell, 2022).

Deep learning models compute rich, high-dimensional **embeddings** for any given input. Embeddings can then be used for **classification** tasks, for example providing predictions on species presence. Recent studies show that while models are trained to classify specific species, using these models to organize large acoustic datasets based on the extracted embeddings offers up a plethora of opportunities for other bioacoustic applications, for example clustering unlabelled recordings, individual identification or novel call-type detection (Allen-Ankins et al., 2025; Ghani et al., 2024, 2025; Miron et al., 2025). Furthermore, using deep learning embeddings rather than model predictions provides a continuous space where all data are represented rather than relying on model predictions to filter datasets. Models are being trained on broader datasets, going from bird-focused training sets like *BIRB* (Hamer et al., 2023) and the annotated bird recordings on the citizen science platform *xeno-canto* (xeno-canto, 2025) to incorporate more species groups like amphibians (Cañas et al., 2023), mammals (Schäfer-Zimmermann et al., 2024) and insects (Chasmai et al., 2024). Models trained on general audio are also being trained for bioacoustic datasets (Miron et al., 2025; Schwinger et al., 2025). Yet, at this point, clear tendencies on what models to use for an acoustic environment are inconclusive and require researchers to have the tools to test and compare models in order to identify the best method for their research needs.

New bioacoustic **deep learning models** (see Glossary) are trained and released at increasing rates (Miron et al., 2025; Schwinger et al., 2025; Stowell, 2022). At the same time, means to compare and use models are often limited by how well their respective code is maintained. Computing acoustic indices and unsupervised classification of soundscapes have been made accessible through numerous Python software packages, for example for acoustic indices, *scikit-maad* (Ulloa et al., 2021) and for unsupervised classification, *bambird* (Michaud et al., 2023), *soundscape explorer* (Sound-Scape-Explorer/Sound-Scape-Explorer, 2026) and *pykanto* (Recalde, 2023). Using pretrained bioacoustic classification models and comparing them is challenging due to different model requirements. As a result, there is a lack of accessibility in computational bioacoustics which restricts researchers: 1. Deep learning models are primarily used by users with a background in coding, 2. researchers are unable to investigate the usefulness and compare between models for their own data and 3. users have no way to interact with the model outputs. Packages like *OpenSoundscape* (Lapp et al., 2023) and *avex* (Miron et al., 2025) address this gap and provide a variety of other automated processing techniques for soundscape recordings. These packages incorporate a large variety of methods with their main focus being the support of the state-of-the-art in deep learning for bird sound classification. Executing these packages and using their outputs is built around well maintained and well documented Python application programming interfaces (APIs). The question of which deep learning classifiers perform best has been addressed in multiple recent reviews (Miron et al., 2025; Schwinger et al., 2025; van Merriënboer et al., 2024). However, reproducing these results and comparing between models with different audio input lengths remains cumbersome. Furthermore, usage and evaluation of models requires advanced python coding. For example, *BirdNET*, a model trained on a large corpus of bird vocalizations, is available with a graphical user interface (GUI) tool, making it accessible which explains its widespread use by ornithologists and ecologists (Kahl et al., 2021; Pérez-Granados, 2023). As the field of computational bioacoustics is an interdisciplinary field with researchers coming from varying degrees of ecological/computational backgrounds, the accessibility of newly developed methods to researchers with different skill sets, directly impacts if these methods get included in ecological workflows or not.

Here, we present *bacpipe* (bioacoustic collection pipeline), a Python software package to facilitate the use of state-of-the-art bioacoustic deep learning models. By centring *bacpipe* on extracting acoustic representations using deep learning models (generating embeddings), large acoustic datasets can be structured and investigated without limiting the models to their classifier predictions. Thereby, generating embeddings for a variety of models can be incorporated into existing workflows and models can be compared. As a stand-alone software, *bacpipe* provides interactive visualizations of acoustic representations (embeddings) from different models in a dashboard GUI enabling users to explore large datasets and benefit from the structuring capabilities of state-of-the-art deep learning models. As an add-on, the package comes with a variety of evaluation tools enabling researchers to **probe** model performance on their own data. *Bacpipe* is built in a modular design ensuring that with further development, new deep learning models can be easily included. In the following, we will describe how *bacpipe* enables researchers to

1. process large PAM datasets with a variety of state-of-the-art deep learning models,
2. interactively explore acoustic representations visually and aurally,
3. extract classifier predictions from a variety of models and
4. use bacpipe to evaluate models on alternative bioacoustic tasks.

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Glossary
Key vocabulary used in this manuscript with their specific meaning in the field of computational bioacoustics.
Deep learning model: Deep neural network machine learning model based on (for example) convolutional neural network (CNN) or transformer architecture and in this manuscript trained using acoustic recordings. Deep learning models usually consist of feature extractors and classifiers, the former of which is used to generate embeddings. In this manuscript, deep learning model refers to the feature extraction part of the model.
Embeddings: High-dimensional feature vectors created by deep learning models based on a section of audio.
Embedding space: High-dimensional vector space containing embeddings of a model. Dimensionality reduction tools can be used to visualize these spaces.
Classification: Class predictions are generated by the deep learning model by first computing an embedding and subsequently mapping that embedding onto a list of classes.
Benchmarking: Evaluation of deep learning models including their pretrained classifiers.
Clustering: Unsupervised algorithm that organizes the embeddings into clusters, used here to evaluate deep learning models.
Probing: Supervised algorithm used to evaluate deep learning models by fitting a classifier on top of the model.
Linear probing: Training a linear classifier consisting of a single fully connected layer.
kNN probing: Fitting a k nearest neighbour classifier. Using a fraction of the data to fit the classifier and using it to classify the remainder.

2 | BASIC WORKFLOW

Bacpipe can be used via its API or as a stand-alone software. The user can therefore decide which of bacpipe's processing steps to run. Figure 1 shows a schematic overview of these processing steps. In the following section, the package concept and workflow will be explained in more detail.

2.1 | Design philosophy

Bacpipe is designed to target two audiences: (1) experienced listeners with expertise in ecology and bioacoustics and perhaps little experience with *Python* and (2) experienced deep learning researchers working in bioacoustics with *Python* experience. To address both audiences, *bacpipe* is available as a fully functional and well-documented stand-alone *GitHub* repository[2] and as a *pip* (*PiPy*) package. Once started, *bacpipe* processes files based on user-defined configurations and visualizes results in an interactive GUI dashboard (see Section 3). Alternatively, *bacpipe*'s embedding generation or evaluation modules can be used via its API and integrated into existing workflows (see Section 4). Loading and processing audio have been developed to work both on the CPUs of laptop computers and GPUs of computers or servers. In terms of processing, *bacpipe* is centred around the computation of embeddings with a large variety of bioacoustic models, as can be seen for the sections Data, Models and Embeddings in Figure 1. *Bacpipe* handles multiprocessing by allocating multiple CPU cores for loading and preprocessing audio files simultaneously before passing them on to the deep learning model. The pipeline has been successfully tested on very large datasets and can handle cancellation and resumption of processing. Processing of audio files is currently restricted to one deep learning model at a time. Add-on features to evaluate embedding spaces, such as linear probing, clustering and visualization are included in the package (see Section 2.5). Furthermore, **benchmarking** of multi-label classifier predictions is possible. Accessibility and modularity are the core design principles of *bacpipe* to enable all researchers and practitioners to access the state-of-the-art deep learning models for their bioacoustic dataset. The models included in the package can be seen in Table 1.

[2]https://github.com/bioacoustic-ai/bacpipe.

2.2 | Simple starting-point

Bacpipe is always executed in the same way, via `python run_pipeline.py`. The prerequisite is a computer with a Python 3.11/3.12 installation (all instructions can be found at https://github.com/bioacoustic-ai/bacpipe). Upon execution, *bacpipe* will process audio files in a specified folder using the selected bioacoustic deep learning models and visualize the model outputs in a GUI (see Section 3), allowing the user to interact with and investigate the results. To run *bacpipe* on their own data, the user has to specify the source path to the audio files and which models to run in the `config` file before executing `python run_pipeline.py`. All further configurations are optional. These configurations are specified in the `bacpipe/config.yaml` file or, when using the Python API, using `bacpipe.config` (see Table A1). More advanced settings can be modified in the `bacpipe/settings.yaml` file (or the attribute, for API). Configuration is divided across two files, `config` and `settings`, to keep the most common configurations (e.g. modifying the audio source path, choosing what models to run) separate from the advanced settings (e.g. to compute on a GPU rather than the default CPU, to process only annotated segments). This way the user is not required to modify the `settings` file and deal with the large amounts of options. Model-specific configurations can be changed in the respective Python file of each model (for example for BirdNet: 'bacpipe/model_pipelines/feature_extractors/birdnet.py'). For each execution, copies of the `config` and `settings` files are saved in the *logs* folder with a corresponding timestamp, ensuring that users can go back and reproduce previous experiments. Figure 2 shows an example case study using a large unlabelled dataset. Presenting the results in a GUI and enabling

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

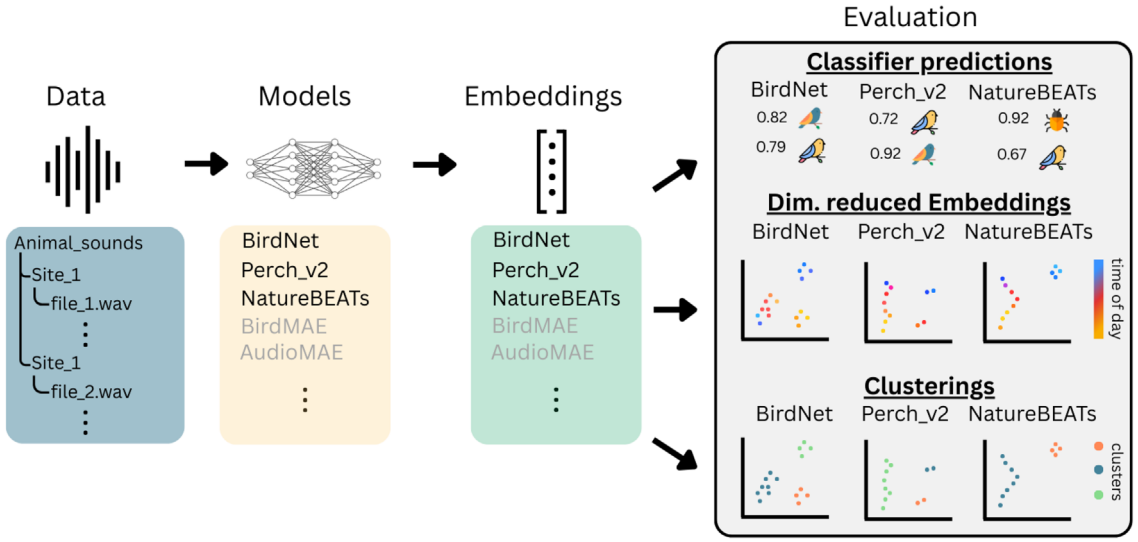


FIGURE 1 Processing workflow of *bacpipe*. From left to right, audio data (Data) are passed into deep neural networks (Models), which generate acoustic feature vectors (Embeddings). Embeddings are then used for a number of tasks which are applied to all selected models for comparability (Evaluation). Evaluation outputs are saved to a standardized folder structure (see Figure 3). By default, evaluations include classification (outputs produce lists of predicted species), dimensionality reduction (used for visualizations) and clustering (used to measure shared mutual information), but can be expanded for other tasks.

both visual and auditory interaction of the processed data helps to make the analysis accessible for all audiences.

2.3 | Reducing unnecessary recomputation

Bacpipe can be used on acoustic datasets of any folder structure or commonly supported audio file formats (*bacpipe* uses `librosa` (McFee et al., 2015) for loading of audio, therefore all formats supported by `librosa` are supported, most common formats include '.wav', '.aif', '.mp3', '.flac', '.ogg'). When processing, *bacpipe* creates a standardized folder structure where embeddings, reduced dimension embeddings, classifier predictions, evaluations and metadata are saved. The top level folder name corresponds to the name of the processed audio dataset. The output folder structure can be seen in Figure 3a. Inside it, the embeddings file structure mirrors the audio file structure, meaning that for every audio file there is a corresponding embedding file; this makes association easier for downstream analysis. The same is true for classifier predictions. The evaluation folder contains classifier predictions, **clustering** results, generated labels and more, which are used for the dashboard GUI and which can also be returned via the API (see Section 4). Saving these processed analyses vastly reduces the computational time for future processing. Once processing is complete, *bacpipe* no longer requires access to the audio files for returning, visualizing or evaluating embeddings, as this is done using the output files. However, interacting with the results and displaying and listening to audio segments corresponding to embedding points does require access to the audio files. By using a standardized output folder structure, results can be transferred between devices enabling one researcher to visualize the results that were processed by another (or to process on a GPU cluster and analyse on a laptop).

High-dimensional embeddings are saved as Numpy array files (ending with `.npy`) to reduce storage requirements, while reduced low-dimensional embedding (by default, 2d-UMAP (McInnes et al., 2020)) files are saved as human-readable dictionary files (ending with `.json`). By default, classification predictions are saved as annotation tables (one per audio file) in the Raven (Cornell Lab of Ornithology, 2014) format as well as a combined annotations file (one .csv file with each row corresponding to a timestamp of a prediction within an audio file) for all classifier predictions in the dataset.

2.4 | Metadata and ground truth labels

Bioacoustic datasets often feature a nested folder structure based on deployment sites and date. Using this information along with the file-specific timestamps can be helpful to analyse species activity and embedding structure in respect to spatial and temporal patterns. To make use of this, *bacpipe* creates metadata labels based on this information. Metadata labels refer to metadata that is extracted from the folder structure and file timestamps based on common naming conventions in PAM data. These metadata labels are associated with embeddings and thereby offer the opportunity to evaluate correlations between embedding structure and metadata both visually and computationally. At the point of writing, the metadata labels that are automatically generated are 'time_of_day', 'week_of_year', 'day_of_year', 'continuous_timestamp', 'parent_directory' and 'audio_file_name'. 'parent_directory' and 'audio_file_name' are inferred based on the file name and location within the folder structure. 'time_of_day', 'week_of_year', 'day_of_year' and 'continuous_timestamp' are all inferred from the file name based on PAM naming conventions (extraction is done using a regular expression). As different models

20412210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 1 List of models currently available in bacpipe.

Name	Clfier	Training	Architecture	Dim	Trained on	sr [kHz]	Size [10^6]	Ref.
AudioMAE	No	ssl+ft	ViT	768	General	16.0	86.0	Huang et al. (2022)
AudioProtoPNet	Yes	supl	ConvNeXt	768	Birds	32.0	98.0	Heinrich et al. (2025)
AvesEcho	Yes	supl	PaSST	1024	Birds	32.0	86.3	Ghani et al. (2024)
AVES	No	ssl+ft	HuBERT	768	General, biotic	16.0	94.2	Hagiwara (2022)
Bat	Yes	supl	CNN+Trafo	64	Bats	220	0.07	Fundel et al. (2023)
BatDetect2_Clip	Yes	supl	Net2DFast	32	Bats	256	5.0	Mac Aodha et al. (2018)
BatDetect2_Dets	No	supl	Net2DFast	32	Bats	256	5.0	Mac Aodha et al. (2018)
BEATs	No	ssl+ft	HuBERT	768	General	16.0	91.0	Chen et al. (2022)
BioLingual	No	supl	CLAP	512	Birds, animals	48.0	190.0	Robinson et al. (2023)
BirdAVES	No	ssl+ft	HuBERT	1024	General, birds	16.0	316.0	Hagiwara (2022)
BirdMAE	No	ssl+ft	ViT	768	General	32.0	300.0	Rauch, Heinrich, et al. (2025)
BirdNET	Yes	supl	EffNetB0	1024	Birds	48.0	12.8	Kahl et al. (2021)
BirdNET_v3	Yes	supl	EffNetv2s	1280	Birds, animals	32.0	27.0	Lasseck et al. (2026)
ConvNeXt_bs	Yes	supl	ConvNeXt	768	Birds	32.0	88.0	Schwinger et al. (2025)
Google_Whale	Yes	supl	EffNetB0	1280	Whales	24.0	5.0	-
HBdet	Yes	supl	ResNet50	1024	Hback. Whales	2.0	23.0	Kather et al. (2024)
Insect459NET	No	supl	EffNetv2s	1280	Insects	44.1	20.6	-
Insect66NET	No	supl	EffNetv2s	1280	Insects	44.1	20.5	-
Mix2	No	supl	MobileNetv3	960	Amphibians	16.0	3.0	Moummad, Farrugia, et al. (2024)
NatureBEATs	No	ssl+ft	HuBERT	768	All	16.0	90.7	Robinson et al. (2024)
Perch_Bird	Yes	supl	EffNetB1	1280	Birds	32.0	7.8	Ghani et al. (2023)
Perch_v2	Yes	supl	EffNetB3	1536	Animals	32.0	12.0	van Merriënboer et al. (2025)
ProtoCLR	No	sup cl	CvT-13	384	Birds	16.0	19.6	Moummad, Serizel, et al. (2024)
RCL_FS_BSED	No	sup cl	ResNet9	2048	Animals	22.0	7.2	Moummad, Serizel, & Farrugia (2024)
SurfPerch	Yes	supl	EffNetB0	1280	Corals, birds	32.0	8.0	Williams et al. (2024)
VGGish	No	supl	VGG	128	General	16.0	62.0	Hershey et al. (2017)

Note: The column headings correspond to the following: **Clfier** lists if a classifier is included for the model, **Training** shows the training setup, that is *ssl* for self-supervised learning, *supl* for supervised learning, *sup cl* for supervised contrastive learning and *ft* for fine-tuning. **Architecture** lists the model backbone and **Dim** shows the dimension of the embeddings. **Trained on** shows the model's training data, **sr** lists the sample rate in kHz, size shows the model **Size** in millions of parameters and **Ref.** lists the respective publication.

use different input lengths of audio, the metadata labels get mapped onto the model-specific timestamps, thereby associating each embedding with a default label.

During each execution, *bacpipe* will also search for ground truth labels in the form of a ground truth annotations file named 'annotations.csv' (can be changed in the `settings` file) in the root directory of the audio data. If this file does not exist, the program continues normally. If it does find this file, it requires specific column names to be parsed: 'audiofilename', 'start', 'end' and 'label:species' (species can be replaced, but 'label:' needs to remain to clarify that this is a label column). Users can add multiple 'label:...' columns, thereby supporting annotations of different granularities, for example species, genus, individual. Once the ground truth annotations are read, the labels are mapped onto the timestamps of models and within the `evaluations/labels` folder, ground truth files are created for each existing 'label:' column (in the resulting files, different label categories are columns and annotation events are rows). Standardizing the ground truth labels in this way makes them available in the right format for bacpipe to be used for benchmarking, evaluating clustering, visualizing or linear probing.

2.5 | Benchmarking, clustering and probing

Traditionally, deep learning models are evaluated using benchmarking, in which classifier predictions are generated for a benchmarking dataset. These benchmarking datasets must not be included in the

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

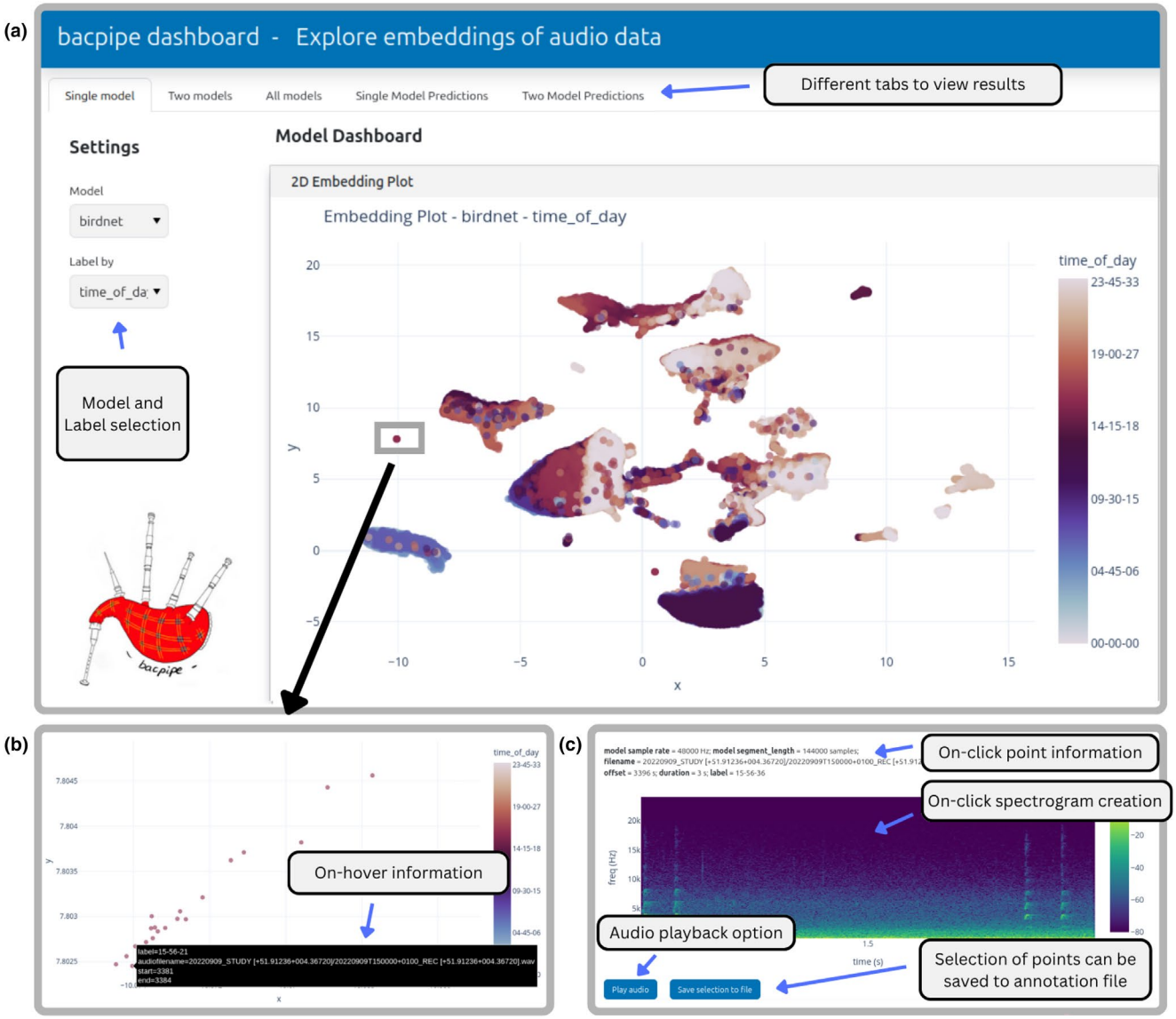


FIGURE 2 A large unlabelled dataset from a noisy urban soundscape in the Netherlands is visualized in different ways. (a) *Bacpipe* dashboard view of a 2d UMAP visualization of BirdNET embeddings from the entire dataset. The embeddings ($\approx 10^5$) were processed from 104 recordings (≈ 100 h of audio). Little boxes indicate what purpose different sections of the dashboard serve. The points are coloured by generated time-of-day timestamps. While the view of the entire dataset (a) shows some structure, the grey rectangle corresponds to the zoomed-in view (b). In (c) a spectrogram is shown visualizing the audio data corresponding to a selected embedding point in (b). Using the inbuilt functions, a handful of unknown bird vocalizations could be found and isolated, listened to and exported as a .csv file to isolate them from the otherwise anthropogenic noise-dominated soundscape. Without any ground truth, these processing and interactive capabilities make evaluating large noise-dominated datasets more feasible.

training data. Ideally, bioacoustic benchmarking datasets feature data from different environments, so that the generalization capabilities of the models can be tested. The resulting predictions are evaluated, for example using mean average precision (mAP) and compared against the state-of-the-art to showcase improvements. Common bioacoustic benchmarking datasets for bird classification are BEANS (Hagiwara et al., 2023) for single label or BIRB (Hamer et al., 2023) and BirdSet (Rauch, Schwinger, et al., 2025) for multi-label classifier performance. With the publication of InsectSet459 (Faiß et al., 2025) and AnuraSet (Cañas et al., 2023), deep learning models can now also be evaluated on different species groups (Kather et al., 2025a).

Bacpipe does not include any datasets. However, it facilitates benchmarking by preparing and evaluating the data for each model: It loads the annotations, the audio corresponding to the annotations and pads the audio lengths to correspond to the model-specific input lengths (the padding strategy can be specified in the `settings` file). Once the audio data are processed, predicted species are compared to ground truth labels and *scikit-learn* (Pedregosa et al., 2011) functions are used to produce performance metrics. Annotations have to be collected in a single `.csv` file as described in Section 2.4. Species names in the annotations have to correspond to the species names that are used in

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

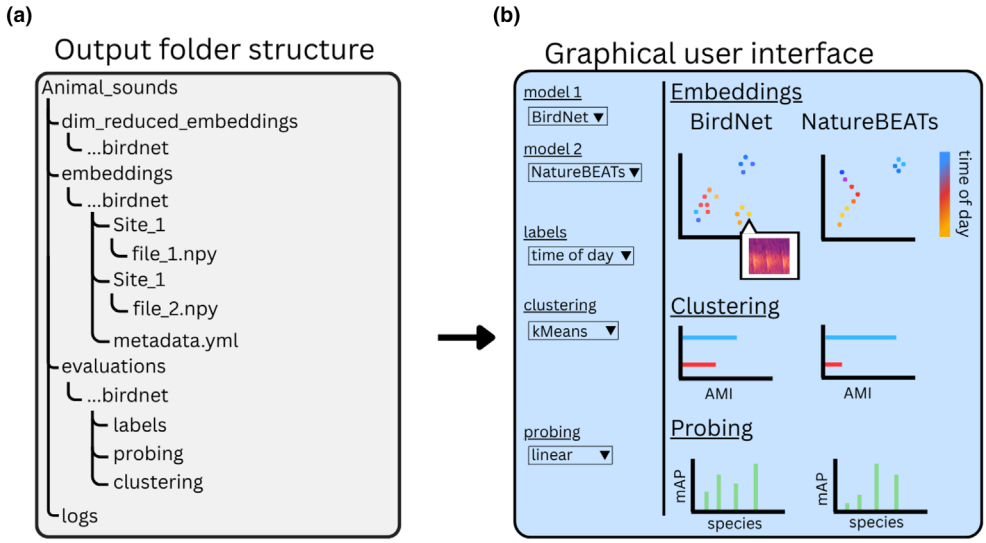


FIGURE 3 Output folder structure (a) and graphical user interface (b) of *bacpipe*. In (a) the standardized folder structure is shown. This is how the processed files are saved, showing that the saved embeddings mirror the input structure from Figure 1. Furthermore, evaluations, dimensionality-reduced embeddings, and processing logs are saved. In (b) the graphical interface that is produced by default is shown. The interface enables evaluating the results in an accessible way.

deep learning classifiers, which are the species common name or their respective species eBird (Sullivan et al., 2009) code. *Bacpipe* will produce print messages if species in the annotations were not found in the deep learning classifiers species list. To run the benchmark, users have to use the API function `bacpipe.benchmark()`. The function requires the user to specify the dataset name corresponding to the top level audio folder (see Section 2.3) and the model they wish to evaluate. To provide an example: users can use the dataset published by Clapp et al. (2023). After downloading the dataset, users have to simply change the column names of the 'annotations.csv' file to be in accordance with the expected column names (see Section 2.4): 'Filename' to 'audiofilename', 'Start Time (s)' to 'start', 'End Time (s)' to 'end' and 'Species eBird Code' to 'label:species'. After applying these changes and selecting the models to benchmark, *bacpipe* will compute embeddings and classifier predictions and evaluate their performance.

To quantify a deep learning model's ability to extract acoustic features, recent research has moved beyond the traditional benchmarking of classifier performance against a single test set. Instead, to evaluate **embedding spaces**, new classification heads are trained and evaluated on top of pretrained models—this process is called probing (see glossary) (Alain & Bengio, 2018). In the following, probing refers to the process of training **kNN** or **linear probes** on top of feature extractors, while classification refers to the use of the integrated classification head which is part of a published deep learning model. Recent studies have applied this practice to bioacoustics, enabling them to compare models that were trained on different species (Kather et al., 2025b; Miron et al., 2025; Schwinger et al., 2025; van Merriënboer et al., 2025). *Bacpipe* includes both training linear and kNN probes on top of processed embeddings. A linear probe is a trainable single fully connected layer which can be useful to test the global structure of the embeddings by trying to separate classes linearly. A kNN probe on the other hand is a classifier based on (Euclidean) distance, which emphasizes local structure by testing if nearest neighbour associations are useful to separate classes. These evaluation tasks do not require access to the models, as they can be processed on the precomputed embeddings.

Another evaluation strategy supported by *bacpipe* is clustering. Past studies have shown that KMeans clustering provides a fast and scalable way to evaluate bioacoustic embedding spaces (Kather et al., 2025b). Clustering can be enabled by adding 'clustering' to the evaluation tasks in the `config` file. If clustering is enabled, by default a KMeans clustering of the embeddings (in their original high dimension) is computed. The clustering is evaluated by comparing the cluster identities of embeddings to different labels and measuring the shared information. This is quantified using adjusted mutual information (AMI) (Romano et al., 2014) and adjusted Rand index (ARI) (Steinley et al., 2016). Cluster allocations are compared to the automatically generated metadata labels (see Section 2.4). Therefore, clustering can be computed and evaluated without ground truth annotations. However, if ground truth annotations are provided, the clustering is also evaluated against it, both for the entire dataset and once for only the annotated portion of the dataset. This way embeddings corresponding to vocalizations of specific species can be evaluated disregarding periods of 'noise'.

If the probing evaluation task is enabled (by adding 'probing' to the evaluation tasks in the `config` file), *bacpipe* requires ground truth annotations (see Section 2.4). The user has to simply specify which evaluation tasks to run in the `config` file, and subsequently, these evaluations can be immediately computed for each of the selected models. Parameters for these evaluations

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License



FIGURE 4 Case study of a large annotated dataset using AnuraSet which consists of recordings from the Amazonian rainforest (Cañas et al., 2023). In (a) a *bacpipe* dashboard is shown with a side-by-side comparison of 2d UMAP visualizations of embeddings generated using two different models: AvesEcho_Passt (Ghani et al., 2025) and NatureBEATs (Robinson et al., 2024). Little boxes indicate what purpose different sections of the dashboard serve. The points are coloured by the parent directory of the audio file they were processed from, which corresponds to the recording site. The embedding plot for NatureBEATs shows fewer points, as only the annotated portion of the dataset was chosen to be displayed. For (b) the model was retrained using the probing procedure (see Section 2.5) on species which were annotated in the AnuraSet dataset. With the linear probe, the model is now able to classify frog species and the results of the class-wise performance are shown in a bar plot. In (c) an automatically generated presence heatmap is shown of the frog species *Sphaenorhynchus surdus* using the model AvesEcho_Passt with the trained linear probe. These heatmaps can be generated for all species present in the annotations. This workflow demonstrates that *bacpipe* can be used to retrain models on classes they were not originally trained on and visualize species presence immediately.

have default values and so can be computed immediately, but can also be customized by referring to the `settings` file. To train linear and kNN probes on the embeddings, the annotations are split into train, validation and test. Results are evaluated per label class and as a global average using mean average precision (see Figure 4).

3 | GRAPHICAL USER INTERFACE

Bacpipe generates a browser-hosted graphical user interface based on the Python package *panel*.[3] A schematic of this is shown in

[3] https://panel.holoviz.org/.

20412108x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Figure 3b. The interface shows different tabs, which visualize the results in two strategies, 1. display model-specific embedding spaces and 2. display model-specific heatmaps of classifier predictions. For each visualization, there are tabs showing the performance for 1. a single model and 2. a side-by-side comparison of two models. For the embedding visualizations, there is an additional overview of all models. In the following, the visualization strategies are explained in detail. Figures 2 and 4 show two case studies using the graphical user interface.

3.1 | Explore visually and aurally

While deep learning models enable the processing of vast amounts of audio data, the interpretability of their output can be limited. Classification performance of models is steadily increasing, yet, many species of interest are not well represented in datasets (or not represented at all), leading to classifier predictions being irrelevant for a given research project. Beyond classifier predictions, it is also possible to visualize embeddings produced by deep learning models using dimensionality reduction techniques (see Section 4.2). The vast amounts of embeddings show emerging structures and clusters that, when overlaid with labels, reveal patterns of how data is organized by models (for more information on labels see Section 2.3). This way, visualizations of embeddings can be colour-coded with automatically extracted metadata labels like timestamps, as well as computed clusters or (if provided) ground truth labels (see Section 2.4). Side-by-side comparisons of models make differences apparent and enable users to decide based on their study species or noise environment which model is best suited. For ecoacoustics, this comparison can be helpful to investigate how differences between soundscapes relate to differences in space, time or habitat characteristics.

Low-dimensional (2 or 3-dimensional) UMAP (or t-SNE) embeddings have the benefit that they can be visualized. However, without the ability to link the embeddings back to the underlying audio source, evaluations are limited to qualitative differences in appearance (McInnes et al., 2020). We therefore provide interactive visualizations based on the Python package *plotly* (Sievert et al., 2015), which displays a spectrogram of the audio when clicking on an embedding point. The spectrogram's audio can also be played back. This connects the high-level deep learning embeddings back to its source and enables an intuitive exploration of audio data while benefiting from the structuring and processing capabilities of deep learning. As the spectrograms are generated in real-time, access to the original audio data is required. Audio is resampled to the model-specific sample rate. The maximum frequency of the visualized spectrogram corresponds to half of the sample rate, as follows from the Nyquist criterion. This way the user is provided with visual feedback of what portion of the spectrogram a model 'saw' to compute the subsequent embedding. Figure 2 shows a case study in which embeddings and spectrogram visualizations were used to isolate bird vocalizations in a large unlabelled dataset.

By using *plotly*, users can zoom and pan in the embedding space. For PAM datasets spanning years of data, this is essential, as the large number of embeddings contains complex structures and substructures that only become apparent when narrowing in on small sections. A selection tool can be used to choose a number of points in the embedding space, which can then be exported as annotations to a csv file. This can be useful for the task of annotating or investigating segments with similar acoustic characteristics.

3.2 | Visualize clustering and probing metrics

In addition to the embedding visualization, the GUI provides extra charts to visualize the metrics specific to clustering and probing (if enabled in config file). The way in which clustering and probing are computed and evaluated is explained in Section 2.5. If ground truth data is provided, the clustering is also computed between the metadata labels (see Section 2.4) and the ground truth. This can show the overlap between the presence of classes and diurnal, seasonal or habitat-based patterns. Figure 4a shows a side-by-side comparison of two models, in which points are labelled by the parent directory. For the dataset used in this example, AnuraSet (Cañas et al., 2023), the parent directory of an audio file corresponds to the location site. The model *naturebeats* (on the right) shows a strong organization of points according to the parent directory. While Figure 4 focuses on the probing features, *bacpipe's* clustering capabilities could be used to evaluate the AMI and ARI values between the clustering and the parent directory. Alternatively, the overlap between the site and the ground truth could be quantified to indicate if species only vocalize in specific places.

If probing is enabled in the configurations, kNN and linear probes are trained on the provided ground truth annotations (as described in Section 2.5). The results of the probing are shown as bar plots. Users can switch between kNN and linear probes to see the respective results. For each label class, the accuracy values are displayed. Using the probing evaluation, users can see how well a model's embeddings can be classified into the annotated species using both kNN and linear probing. Figure 4b shows the class-wise performance of the AvesEcho_Passt model using linear probing on the AnuraSet dataset. For both the clustering and the classification, overall performance is displayed in the all_models tab.

3.3 | Classification and probing heatmaps

While generating embeddings, by default *bacpipe* also generates classifier predictions using the model provided classifier. These predictions are saved as Raven style annotations (see Section 2.3). The generated predictions are displayed in the dashboard of *bacpipe* as an activity heatmap. The heatmap is generated to show the dates

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

on the y- and the hours on the x-axis and colour-coded by the number of occurrences. Users can choose between species from a drop-down menu and thereby visualize the activity for each species found within the dataset.

As discussed in Section 2.5, probing can be applied to classify the embeddings. In this case, the previously trained linear probe is loaded and used to classify the precomputed embeddings. This way a classifier for a new task (e.g. different species group, individual identification) is trained and can be used to generate predictions. For models that do not provide classification layers (see Table 1), this provides an opportunity to generate classifier predictions. The results of this classification can also be displayed in the same way as the pretrained classifiers. Figure 4c shows a species presence heatmap which was generated using a linear probe trained on the model AvesEcho_Passt using frog vocalizations annotated in the AnuraSet dataset (Cañas et al., 2023).

4 | DEVELOPER SECTION

For users that look to integrate *bacpipe* into their existing bioacoustic pipelines, an API is provided. This API features the same modules and functions as described in Section 2. In the following section, the API will be described in more detail. For more information, see the documentation.[4]

4.1 | Package structure

The package is made up of three subpackages: `core`, `model_pipelines` and `embedding_evaluation`. The subpackages handle audio loading and preparing inference (`core`), loading and executing the respective deep learning models (`model_pipelines`) and evaluating the generated embeddings through linear probing, classification and visualization (`embedding_evaluation`). *Bacpipe* saves embeddings as *NumPy* (Harris et al., 2020) files (ending on `.npy`) so that each audio file has a corresponding *NumPy* file with the same file name and the appendix '_model-name', where model-name corresponds to the name of the deep learning model used to process the audio. Upon execution, `core` checks to see if the embeddings generated by the combination of input data and selected deep learning models already exist. This is done by comparing the file names of audio files and the created *NumPy* files. `embedding_evaluation` is used to load visualizations and evaluations of the data within seconds rather than requiring to be computed again. This also allows continuously growing datasets or prematurely failed processing runs to be continued where left off, which is practical when processing very large PAM datasets. Throughout the processing, a (human-readable) metadata file (`metadata.yml`) is created with detailed information within the corresponding embeddings folder. All of these convenience features can be activated or deactivated using the Python API.

[4] https://bacpipe.readthedocs.io.

4.2 | Extending *bacpipe*

The aim of *bacpipe* is to streamline the processing of audio recordings using interchangeable bioacoustic deep learning models. The models included in the package can be seen in Table 1. Once the brief installation of *bacpipe* is successful, the user will be able to process their audio data using a plethora of different deep learning models, among them the most used and state-of-the-art models. Comparing existing models against newly developed ones is crucial, which is why the modular design of the package allows users to easily add new models or add modified versions of existing ones. Any new model will go through the same evaluation pipeline, this way the results of new/modified models can be explored and compared. To visualize the model outputs, dimensionality reduction techniques (e.g. UMAP (McInnes et al., 2020), t-SNE (Van der Maaten & Hinton, 2008), PCA (Wold et al., 1987)) are included in *bacpipe*. Generated visualizations can be used to compare how different models structure the data and are interactive, allowing the user to connect the embeddings back to the audio segment they were processed on. Just like bioacoustic deep learning models, they too can be added or modified for comparisons. Finally, the evaluation tool-set are functions applied to all specified models and this tool-set can also be changed or expanded as the field advances. The section Evaluation in Figure 3 shows some of these downstream processing steps.

4.3 | Low to high level pipelines

Bacpipe comes with a high-level prepackaged fully operational pipeline, accessible through `bacpipe.play()`. Based on the attributes of `bacpipe.config` and `bacpipe.settings`, all audio files in a given directory will be processed with all selected models. Subsequent evaluation will be performed, and visualization of the results is facilitated in the dashboard GUI (explained in Section 3). However, to integrate smoothly into existing workflows, numerous pipelines are provided to work with varying levels of underlying processing. This way users can run a pipeline that will process audio folders for 1. all specified models, 2. a single specified model or 3. a single audio file for a given model. A selection of API functions along with descriptions are provided in the Appendix (see Table A1).

The lower level workflow of *bacpipe* divides the main tasks of path-handling, audio-handling, model-handling and classifier-handling into the four classes: `Loader`, `AudioHandler`, `Embedder` and `Classifier`. To initialize the processing, `bacpipe` uses the *Loader* class, which organizes the loading of audio files and embedding files (if already processed). The `AudioHandler` class handles loading of audio files itself, along with resampling to model-specific sampling rates, padding audio and segmenting it into batches. The batched file segments then get passed on to the `Embedder` class, which handles loading the model and generating embeddings. Finally, the `Classifier` class receives embeddings and generates class predictions. As the `Loader` class handles all paths, the object generated from it can be used to load all embeddings, predictions and metadata after processing.

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

A large number of functions are provided to facilitate the integration into the different *bacpipe* pipelines. These functions can be used to extract date-time information from audio files, generate arrays of ground truth labels fitted to the model-specific timestamps, list all processable audio files in directories and more. Higher level functions allow users to generate clustering, train and evaluate probes and visualize their results. Finally, a benchmarking function allows users to evaluate single or multi-label model predictions in respect to a provided annotation file (see Section 2.5).

5 | PERSPECTIVES

The field of computational bioacoustics is experiencing an accelerated development, fuelled by advancements in deep learning and interest in questions of ecology and evolution. The previous decades have seen the creation of massive corpora of recordings being collected by researchers, conservationists and citizen scientists all over the world. However, evaluating datasets of very large proportions without computational tools is extremely time-consuming. In practice, for this reason, evaluations are often restricted to small portions of the datasets.

Here, we presented *bacpipe*, a collection of bioacoustic pipelines consisting of state-of-the-art deep learning models and evaluation techniques, made accessible through a graphical and a programming interface. *Bacpipe* is inspired by and builds on the variety of tools, which have been developed and published for the processing of bioacoustic data (mentioned in Section 1). However, it is the first tool to include all the state-of-the-art models in bioacoustics and make them accessible to both computer scientists and ecologists. Its modular setup is designed to integrate with existing tools and workflows to ensure researchers can compare models for their specific hypotheses. Tools like *OpenSoundscape* (Lapp et al., 2023) and *avex* (Miron et al., 2025) have proposed workflows of processing audio recordings using deep learning models. *Bacpipe* is another contribution in this field focusing on integrating a wide range of deep learning models and making them accessible through functionalities and a GUI. A next step for the field could be trying to converge on common entry points to enable users to combine different packages' capabilities.

With the accelerated pace of deep learning developments and models being released, accessibility of these methods will be increasingly important. At the same time, archives of acoustic datasets all over the world span millions of hours and contain valuable ecological and evolutionary information. By harnessing the full potential of deep learning models, this information can be organized. If combined with interactive tools, these datasets can be explored and isolated events and vocalizations can be discovered.

While recent review papers show that scores are improving for bird song classification, noisy and polyphonic soundscape recordings still lead to poor classifier performance (Miron et al., 2025; Schwinger et al., 2025; van Merriënboer et al., 2025). We argue that the usage of deep learning models in bioacoustics should not be limited to classifier predictions. Instead, using deep learning models as acoustic feature extractors to organize and reveal structure in large PAM datasets empowers users to explore and identify sounds of interest through interactivity. Figure 2 shows a case study of this workflow.

Computational bioacoustics is an interdisciplinary field that requires interdisciplinary workflows to best combine people's expertise from their respective fields. Like many before it, the aim of *bacpipe* is to be a collaborative tool, shaped by and for the community. Contributions are very welcome, especially newly developed models by deep learning practitioners and evaluation workflows by ecologists. We refer to the GitHub repository for more detailed descriptions.[5] We hope that this introduction and description will invite researchers to collaborate and design systems accessible to all audiences.

AUTHOR CONTRIBUTIONS

Vincent S. Kather: Conceptualization, software, methodology, writing—original draft, writing—review and editing. **Sylvain Haupert:** Methodology, writing—review and editing, supervision. **Burooj Ghani:** Methodology, writing—review and editing, supervision. **Dan Stowell:** Methodology, writing—review and editing, supervision, funding acquisition. All authors contributed critically to the drafts and gave final approval for publication.

ACKNOWLEDGEMENTS

The authors would like to thank all people that have contributed issues, pull requests, feedback and especially the creators of the deep learning models. The authors would like to specifically thank Minkyung Kwak for her valuable feedback and testing of the bacpipe GUI. The authors would like to also thank Nicole Allison for the design of the *bacpipe* logo.

FUNDING INFORMATION

VSK, SH and DS are supported by Marie Skłodowska-Curie Action Bioacoustic AI (grant no. 101116715). BG is supported by the Horizon Europe projects TETTRIs (grant no. 101081903) and MAMBO (grant no. 10106063).

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

PEER REVIEW

The peer review history for this article is available at https://www.webofscience.com/api/gateway/wos/peer-review/10.1111/2041-210x.70406.

DATA AVAILABILITY STATEMENT

Bacpipe is developed openly in GitHub at https://github.com/bioacoustic-ai/bacpipe, available under Apache 2.0 open-source licence. Documentation is available at https://bacpipe.readthedocs.io. *Bacpipe* is installable through PyPi (https://pypi.org/project/bacpipe/) and has been tested on Linux, Mac and Windows. The

[5]https://github.com/bioacoustic-ai/bacpipe.

20412210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

bacpipe version corresponding to this manuscript is v1.3.5 and is available at [https://doi.org/10.5281/zenodo.22142045] (Kather et al., 2026).

ORCID

Vincent S. Kather https://orcid.org/0000-0003-1491-0964
Sylvain Haupert https://orcid.org/0000-0003-4705-4527
Burooj Ghani https://orcid.org/0000-0002-0541-3571
Dan Stowell https://orcid.org/0000-0001-8068-3769

REFERENCES

Alain, G., & Bengio, Y. (2018). Understanding intermediate layers using linear classifier probes.

Allen-Ankins, S., Hoefer, S., Bartholomew, J., Brodie, S., & Schwarzkopf, L. (2025). The use of BirdNET embeddings as a fast solution to find novel sound classes in audio recordings. *Frontiers in Ecology and Evolution*, *12*. https://doi.org/10.3389/fevo.2024.1409407

Baumgartner, M. F., & Mussoline, S. E. (2011). A generalized baleen whale call detection and classification system. *Journal of the Acoustical Society of America*, *129*(5), 2889–2902. https://doi.org/10.1121/1.3562166

Calupca, T. A., Fristrup, K. M., & Clark, C. W. (2000). A compact digital recording system for autonomous bioacoustic monitoring. *The Journal of the Acoustical Society of America*, *108*(5_Supplement), 2582. https://doi.org/10.1121/1.4743595

Cañas, J. S., Toro-Gómez, M. P., Sugai, L. S. M., Benítez Restrepo, H. D., Rudas, J., Bautista, B. P., Toledo, L. F., Dena, S., Domingos, A. H. R., de Souza, F. L., Neckel-Oliveira, S., da Rosa, A., Carvalho-Rocha, V., Bernardy, J. V., Moreira Sugai, J. L. M., dos Santos, C. E., Bastos, R. P., Llusia, D., & Ulloa, J. S. (2023). AnuraSet: A dataset for benchmarking Neotropical anuran calls identification in passive acoustic monitoring.

Chasmai, M., Shepard, A., Maji, S., & Van Horn, G. (2024). The inaturalist sounds dataset. *Advances in Neural Information Processing Systems*, *37*, 132524–132544.

Chen, S., Wu, Y., Wang, C., Liu, S., Tompkins, D., Chen, Z., & Wei, F. (2022). BEATs: Audio Pre-Training with Acoustic Tokenizers.

Clapp, M., Kahl, S., Meyer, E., McKenna, M., Klinck, H., & Patricelli, G. (2023). A collection of fully-annotated soundscape recordings from the southern Sierra Nevada mountain range. Dataset on Zenodo.

Cornell Lab of Ornithology. (2014). Raven Pro: Interactive Sound Analysis Software. Version 1.5.

Desjonquères, C., Linke, S., Greenhalgh, J., Rybak, F., & Sueur, J. (2024). The potential of acoustic monitoring of aquatic insects for freshwater assessment. *Philosophical Transactions of the Royal Society, B: Biological Sciences*, *379*(1904), 20230109. https://doi.org/10.1098/rstb.2023.0109

Faiß, M., Ghani, B., & Stowell, D. (2025). InsectSet459: An open dataset of insect sounds for bioacoustic machine learning.

Fundel, F., Braun, D. A., & Gottwald, S. (2023). Automatic bat call classification using transformer networks. *Ecological Informatics*, *78*, 102288. https://doi.org/10.1016/j.ecoinf.2023.102288

Ghani, B., Denton, T., Kahl, S., & Klinck, H. (2023). Global birdsong embeddings enable superior transfer learning for bioacoustic classification. *Scientific Reports*, *13*(1), 22876. https://doi.org/10.1038/s41598-023-49989-z

Ghani, B., Kalkman, V. J., Planqué, B., Vellinga, W.-P., Gill, L., & Stowell, D. (2024). Generalization in birdsong classification: Impact of transfer learning methods and dataset characteristics.

Ghani, B., Kalkman, V. J., Planqué, B., Vellinga, W.-P., Gill, L., & Stowell, D. (2025). Impact of transfer learning methods and dataset characteristics on generalization in birdsong classification. *Scientific Reports*, *15*(1), 16273. https://doi.org/10.1038/s41598-025-00996-2

Hagiwara, M. (2022). AVES: Animal Vocalization Encoder based on Self-Supervision.

Hagiwara, M., Hoffman, B., Liu, J. Y., Cusimano, M., Effenberger, F., & Zacarian, K. (2023, June). BEANS: The benchmark of animal sounds. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1–5). IEEE.

Hamer, J., Triantafillou, E., van Merriënboer, B., Kahl, S., Klinck, H., Denton, T., & Dumoulin, V. (2023). BIRB: A Generalization Benchmark for Information Retrieval in Bioacoustics.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Heinrich, R., Rauch, L., Sick, B., & Scholz, C. (2025). AudioProtoPNet: An interpretable deep learning model for bird sound classification. *Ecological Informatics*, *87*, 103081. https://doi.org/10.1016/j.ecoinf.2025.103081

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., & Wilson, K. (2017). CNN Architectures for Large-Scale Audio Classification.

Huang, P.-Y., Xu, H., Li, J., Baevski, A., Auli, M., Galuba, W., Metze, F., & Feichtenhofer, C. (2022). Masked autoencoders that listen. *Advances in Neural Information Processing Systems*, *35*, 28708–28720.

Kahl, S., Wood, C. M., Eibl, M., & Klinck, H. (2021). BirdNET: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, *61*, 101236. https://doi.org/10.1016/j.ecoinf.2021.101236

Kather, V., Haupert, S., Parcerisas, C., Martinez Balvanera, S., Nolasco, I., Ben, M. E., Diemo, S., & Dubois, L. (2026). Bioacoustic-ai/bacpipe: BacPipe v1.3.4 Release Notes. https://doi.org/10.5281/zenodo.22035529

Kather, V. S., Ghani, B., & Stowell, D. (2025a). Clustering and novel class recognition: Evaluating bioacoustic deep learning feature extractors.

Kather, V., Ghani, B., & Stowell, D. (2025b). Clustering and novel class recognition: Evaluating bioacoustic deep learning feature extractors. In *Forum acusticum Euronosie 2025: 11th Convention of the European Acoustics Association*, Málaga, June 2025. Forum acusticum Euronosie 2025. https://doi.org/10.61782/fa.2025.0231

Kather, V., Seipel, F., Berges, B., Davis, G., Gibson, C., Harvey, M., Henry, L.-A., Stevenson, A., & Risch, D. (2024). Development of a machine learning detector for North Atlantic humpback whale song. *The Journal of the Acoustical Society of America*, *155*(3), 2050–2064. https://doi.org/10.1121/10.0025275

Lapp, S., Rhinehart, T., Freeland-Haynes, L., Khilnani, J., Syunkova, A., & Kitzes, J. (2023). OpenSoundscape: An open-source bioacoustics analysis package for python. *Methods in Ecology and Evolution*, *14*(9), 2321–2328. https://doi.org/10.1111/2041-210X.14196

Lasseck, M., Eibl, M., Klinck, H., & Kahl, S. (2026). BirdNET+ V3.0 model developer preview. https://doi.org/10.5281/zenodo.20703646

Mac Aodha, O., Gibb, R., Barlow, K. E., Browning, E., Firman, M., Freeman, R., Harder, B., Kinsey, L., Mead, G. R., Newson, S. E., Pandourski, I., Parsons, S., Russ, J., Szodoray-Paradi, A., Szodoray-Paradi, F., Tilova, E., Girolami, M., Brostow, G., & Jones, K. E. (2018). Bat detective—Deep learning tools for bat acoustic signal detection. *PLoS Computational Biology*, *14*(3), e1005995. https://doi.org/10.1371/journal.pcbi.1005995

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. *SciPy*, *2015*(18-24), 7.

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

McInnes, L., Healy, J., & Melville, J. (2020). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.

Mellinger, D. K., Stafford, K. M., Moore, S. E., Dziak, R. P., & Matsumoto, H. (2007). An overview of fixed passive acoustic observation methods for cetaceans. *Oceanography*, *20*(4), 36–45.

Michaud, F., Sueur, J., Le Cesne, M., & Haupert, S. (2023). Unsupervised classification to improve the quality of a bird song recording dataset. *Ecological Informatics*, *74*, 101952. https://doi.org/10.1016/j.ecoinf.2022.101952

Miron, M., Robinson, D., Alizadeh, M., Gilsenan-McMahon, E., Narula, G., Chemla, E., Cusimano, M., Effenberger, F., Hagiwara, M., Hoffman, B., Keen, S., Kim, D., Lawton, J., Liu, J.-Y., Raskin, A., Pietquin, O., & Geist, M. (2025). What Matters for Bioacoustic Encoding.

Moummad, I., Farrugia, N., Serizel, R., Froidevaux, J., & Lostanlen, V. (2024). Mixture of Mixups for Multi-label Classification of Rare Anuran Sounds.

Moummad, I., Serizel, R., Benetos, E., & Farrugia, N. (2024). Domain-Invariant Representation Learning of Bird Sounds.

Moummad, I., Serizel, R., & Farrugia, N. (2024). Regularized Contrastive Pre-training for Few-shot Bioacoustic Sound Detection.

Pedregosa, F., Pedregosa, F., Varoquaux, G., Varoquaux, G., Normalesup Org, Gramfort, A., Gramfort, A., Michel, V., Michel, V., Logilab Fr, Thirion, B., Thirion, B., Grisel, O., Grisel, O., Blondel, M., Prettenhofer, P., Prettenhofer, P., Weiss, R., Dubourg, V., … Cournapeau, D. (2011). Scikit-learn: Machine Learning in Python. MACHINE LEARNING IN PYTHON.

Pérez-Granados, C. (2023). BirdNET: Applications, performance, pitfalls and future opportunities. *Ibis*, *165*(3), 1068–1075. https://doi.org/10.1111/ibi.13193

Rauch, L., Heinrich, R., Moummad, I., Joly, A., Sick, B., & Scholz, C. (2025). Can Masked Autoencoders Also Listen to Birds?

Rauch, L., Schwinger, R., Wirth, M., Heinrich, R., Huseljic, D., Herde, M., Lange, J., Kahl, S., Sick, B., Tomforde, S., & Scholz, C. (2025). BirdSet: A Large-Scale Dataset for Audio Classification in Avian Bioacoustics.

Recalde, N. M. (2023). Pykanto: A python library to accelerate research on wild bird song. *Methods in Ecology and Evolution*, *14*(8), 1994–2002. https://doi.org/10.1111/2041-210X.14155

Robinson, D., Miron, M., Hagiwara, M., & Pietquin, O. (2024). NatureLM-audio: An Audio-Language Foundation Model for Bioacoustics.

Robinson, D., Robinson, A., & Akrapongpisak, L. (2023). Transferable Models for Bioacoustics with Human Language Supervision.

Romano, S., Bailey, J., Nguyen, V., & Verspoor, K. (2014). Standardized mutual information for clustering comparisons: One step further in adjustment for chance. In *Proceedings of the 31st international conference on machine learning* (pp. 1143–1151). PMLR.

Schäfer-Zimmermann, J. C., Demartsev, V., Averly, B., Dhanjal-Adams, K., Duteil, M., Gall, G., Faiß, M., Johnson-Ulrich, L., Stowell, D., Manser, M. B., Roch, M. A., & Strandburg-Peshkin, A. (2024). Animal2vec and MeerKAT: A self-supervised transformer for rare-event raw audio input and a large-scale reference dataset for bioacoustics.

Schwinger, R., Vali Zadeh, P., Rauch, L., Kurz, M., Hauschild, T., Lapp, S., & Tomforde, S. (2025). Foundation models for bioacoustics – A comparative review.

Sebastián-González, E., & Pérez-Granados, C. (2025). Geographic variation in acoustic signals in wildlife: A systematic review. *Journal of Biogeography*, *52*(6), e15116. https://doi.org/10.1111/jbi.15116

Seyfarth, R. M., & Cheney, D. L. (2010). Production, usage, and comprehension in animal vocalizations. *Brain and Language*, *115*(1), 92–100. https://doi.org/10.1016/j.bandl.2009.10.003

Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M., & Despouy, P. (2015). Plotly: Create Interactive Web Graphics via 'plotly.js'.

Sound-Scape-Explorer/Sound-Scape-Explorer.(2026).Sound-scape-explorer.

Steinley, D., Brusco, M. J., & Hubert, L. (2016). The variance of the adjusted Rand index. *Psychological Methods*, *21*(2), 261–272. https://doi.org/10.1037/met0000049

Stowell, D. (2022). Computational bioacoustics with deep learning: A review and roadmap. *PeerJ*, *10*, e13152. https://doi.org/10.7717/peerj.13152

Sueur, J., Farina, A., Gasc, A., Pieretti, N., & Pavoine, S. (2014). Acoustic indices for biodiversity assessment and landscape investigation. *Acta Acustica united with Acustica*, *100*(4), 772–781.

Sugai, L. S. M., Silva, T. S. F., Ribeiro, J. W., Jr., & Llusia, D. (2019). Terrestrial passive acoustic monitoring: Review and Perspectives. *Bioscience*, *69*(1), 15–25. https://doi.org/10.1093/biosci/biy147

Sullivan, B. L., Wood, C. L., Iliff, M. J., Bonney, R. E., Fink, D., & Kelling, S. (2009). eBird: A citizen-based bird observation network in the biological sciences. *Biological Conservation*, *142*(10), 2282–2292. https://doi.org/10.1016/j.biocon.2009.05.006

Teixeira, D., Maron, M., & van Rensburg, B. J. (2019). Bioacoustic monitoring of animal vocal behavior for conservation. *Conservation Science and Practice*, *1*(8), e72. https://doi.org/10.1111/csp2.72

Ulloa, J. S., Haupert, S., Latorre, J. F., Aubin, T., & Sueur, J. (2021). Scikit-maad: An open-source and modular toolbox for quantitative soundscape analysis in python. *Methods in Ecology and Evolution*, *12*(12), 2334–2340. https://doi.org/10.1111/2041-210X.13711

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11).

van Merriënboer, B., Dumoulin, V., Hamer, J., Harrell, L., Burns, A., & Denton, T. (2025). Perch 2.0: The Bittern Lesson for Bioacoustics.

van Merriënboer, B., Hamer, J., Dumoulin, V., Triantafillou, E., & Denton, T. (2024). Birds, bats and beyond: Evaluating generalization in bioacoustics models. *Frontiers in Bird Science*, *3*. https://doi.org/10.3389/fbirs.2024.1369756

Wall, C. C., Haver, S. M., Hatch, L. T., Miksis-Olds, J., Bochenek, R., Dziak, R. P., & Gedamke, J. (2021). The next wave of passive acoustic data management: How centralized access can enhance science. *Frontiers in Marine Science*, *8*. https://doi.org/10.3389/fmars.2021.703682

Williams, B., van Merriënboer, B., Dumoulin, V., Hamer, J., Triantafillou, E., Fleishman, A. B., McKown, M., Munger, J. E., Rice, A. N., Lillis, A., White, C. E., Hobbs, C. A. D., Razak, T. B., Jones, K. E., & Denton, T. (2024). Leveraging tropical reef, bird and unrelated sounds for superior transfer learning in marine bioacoustics.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1), 37–52. https://doi.org/10.1016/0169-7439(87)80084-9

xeno-canto. (2025). Xeno-canto: Sharing wildlife sounds from around the world. https://xeno-canto.org/

How to cite this article: Kather, V. S., Haupert, S., Ghani, B., & Stowell, D. (2026). Bacpipe: A Python package to make bioacoustic deep learning models accessible. *Methods in Ecology and Evolution*, *00*, 1–14. https://doi.org/10.1111/2041-210x.70406

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

APPENDIX A

TABLE A1 Selection of API functions of *bacpipe*. All pipelines can be used with the supported models or models that are passed during runtime. Extensive documentation available on GitHub (https://github.com/bioacoustic-ai/bacpipe), and readthedocs (https://bacpipe.readthedocs.io).

Name	Brief description
config	Most common configurations (paths, model names)
settings	Optional advanced settings (device, thresholds, etc.)
supported_models	List all supported models
Integrated pipelines	
play	Runs the entire bacpipe pipeline, for all specified models and evaluation tasks and launches interactive dashboard
benchmark	Calculates precision, recall and f1 score for a classifier based on provided ground truth annotations
probing_pipeline	Build, train and evaluate classifier on top of embeddings
Embedder	Class to handle model loading and processing
Embedder.get_embeddings_from_model	Returns a single embedding array from an audio file
generate_embeddings	Generates embeddings and classifier predictions for all audio files in the specified directory
run_pipeline_for_models	Generates embeddings, classifier predictions and dimension reduced embeddings from all specified models
ensure_models_exist	Checks if the model files exist, if not downloads them
Functions for data handling	
Loader	Important class to handle all loading and saving
get_audio_files	List all audio files in a directory with supported formats
Loader.embeddings	Returns a Numpy array (or dictionary) of all embeddings
Loader.predictions	Returns a Numpy array (or dictionary or dataframe) of all classifier predictions
AudioHandler	Handles audio loading and preparing for models
AudioHandler.only_load_annotated_segments	Load, resample and window only annotated segments
get_dt_filename	Return array of datetimes corresponding to the time and date in each audio file name
ground_truth_by_model	Returns dataframe with multi-label ground truth associated with the model specific input time length
metadata_labels	Returns dataframe with automatically generated labels

2041210x, 0, Downloaded from https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210x.70406 by Naturalis Biodiversity Center, Wiley Online Library on [14/10/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License