Package 'soundClass'

July 23, 2025

Title Sound Classification Using Convolutional Neural Networks

Version 0.0.9.2

Author Bruno Silva [aut, cre]

Maintainer Bruno Silva <bmsasilva@gmail.com>

Description Provides an all-in-one solution for automatic classification of sound events using convolutional neural networks (CNN). The main purpose is to provide a sound classification workflow, from annotating sound events in recordings to training and automating model usage in real-life situations. Using the package requires a pre-compiled collection of recordings with sound events of interest and it can be employed for: 1) Annotation: create a database of annotated recordings, 2) Training: prepare train data from annotated recordings and fit CNN models, 3) Classification: automate the use of the fitted model for classifying new recordings. By using automatic feature selection and a user-friendly GUI for managing data and training/deploying models, this package is intended to be used by a broad audience as it does not require specific expertise in statistics, programming or sound analysis. Please refer to the vignette for further information.
Gibb, R., et al. (2019) <doi:10.1111/2041-210X.13101>
Mac Aodha, O., et al. (2018) <doi:10.1371/journal.pcbi.1005995>
Stowell, D., et al. (2019) <doi:10.1111/2041-210X.13103>
LeCun, Y., et al. (2012) <doi:10.1007/978-3-642-35289-8_3>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.2

BugReports https://github.com/bmsasilva/soundClass/issues

Imports seewave, DBI, dplyr, dbplyr, RSQLite, signal, tuneR, zoo, magrittr, shinyFiles, shiny, utils, graphics, generics, keras, shinyjs

Depends shinyBS, htmltools

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-29 22:40:02 UTC

Contents

app_label 2
app_model 3
auto_id 5
create_db 6
find_noise 7
import_audio 8
ms2samples 9
spectro_calls 10
train_metadata 11
%>% 12

Index **13**

`app_label` *Shiny app to label recordings*

Description

Shiny app to label recordings. Use this app to visualize your training recordings, create annotations and store them in a sqlite database. The app has a sidebar panel with the following buttons/boxes to input required user data:

1. Create database – if no database exists to store the annotations, use this button to create one
2. Choose database – choose the database to store the annotations
3. Butterworth filter – check box to apply filter and indicate low and high frequencies in kHz to filter the recordings
4. Time expanded – only used in recorders specifically intended for bat recordings. Can take any numeric value. If the recording is not time expanded the value must be set to 1. If it is time expanded the numeric value corresponding to the time expansion should be indicated
5. Choose folder – choose the folder containing the training recordings

After the spectrogram is ploted:

1. Select events by clicking in the spectrogram on the middle of the event of interest (bat call, bird song, etc)
2. Insert the correct label in the "Label" box and add any additional notes in the "Observations" box
3. Press 'Set labels' button to add labels to database
4. Repeat above steps if more than one set of events is present in the recording

5. Press 'Next' button to advance to next recording or pick another recording from the dropdown list

The spectrogram can be zoomed by pressing mouse button and dragging to select an area and then double click on it. To unzoom simply double clicking on the spectrogram without an area selected. To adjust visualization settings, in the top right, the tab "Spectrogram options" can be used to:

- Threshold – minimum intensity values to show in the spectrogram. A value of 100 will typically be adequate for the majority of the recorders
- Window length – moving window length in ms. Smaller windows best suited for short calls
- Overlap – overlap between consecutive windows, higher values give best visualization but lower performance
- Resolution – frequency resolution of the spectrogram

Usage

```
app_label()
```

Value

Starts the shiny app, no return value.

Author(s)

Bruno Silva

`app_model` *Shiny app to fit a model or run a fitted model*

Description

Shiny app to fit a model from training recordings or to run a fitted model to classify new recordings. This app consists of three GUIs, i.e. three main panels, accessible by the tabs at the top:

1. Create train data – create train data from recordings and their respective annotations database
2. Fit model – fit a model from training data
3. Run model – run a fitted model to classify new recordings

1. Create train data:

This panel is used to create train data from recordings and their respective annotations database. The sidebar panel has the following buttons/boxes to input required user data:

- Choose folder – choose the folder containing the training recordings
- Choose database – choose the database with the annotations for the training recordings
- Time expanded – choose the correct time expansion factor, normally only used in recorders specifically intended for bat recordings. Can take the values "auto", 1 or 10. If the recording is in real time the value must be 1. If it's time expanded, the value 10 or "auto" can be selected. If "auto" is selected it is assumed that sampling rates < 50kHz corresponds to a value of 10 and sampling rates > 50kHz to corresponds to a value of 1

- Spectrogram parameters – different typologies of sound events require different parameters for computing the spectrograms. The more relevant are: size (in ms), which should be large enough to encompass the duration of the largest sound event in analysis (not only in the training data but also in novel recordings where the classifiers are to be applied) and moving window (in ms), that should be smaller for shorter sound events (to capture the quick changes in time) and larger for longer sound events (to avoid redundant information). The other parameters are more generalist and the same values can be used for different sound events, as they only change the definition of the images created. Please refer to spectro_calls documentation for further details

After entering the required information press the button "Create training data from labels" to generate the training data that will be used for fitting a model. This object is saved in the folder containing the training recordings with the name "train_data.RDATA".

2. Fit model:

This panel is used to fit a model from training data. The sidebar panel has the following buttons/boxes to input required user data:

- Choose train data – the file "train_data.RDATA" created in the previous panel
- Choose model – a blank model to be fitted. A custom model is provided but must be copied to an external folder if it is to be used. The model path can be obtained by running the following line at the R console: system.file("model_architectures", "model_vgg_sequential.R", package="soundClass") and should be manually copied to a an external folder
- Model parameters – the train percentage indicates the percentage of data that is used to fit the model while the remaining are used for validation, batch size indicates the number of samples per gradient update, the learning rate indicates the degree of the gradient update, early stop indicates the maximum number of epochs without improvement allowed before training stops and epochs indicate the maximum number of epochs to train. Further information can be found in keras documentation https://keras.io/api/

The model is evaluated during fitting using the validation data. After completion, by reaching the maximum epochs or the early stopping parameters, the fitted model, the fitting log and the model metadata are saved to the folder containing the train data with file names: "fitted_model.hdf5", "fitted_model_log.csv" and "fitted_model_metadata.RDATA" respectively.

3. Run model:

This panel is used to run a fitted model to classify new recordings. The sidebar panel has the following buttons/boxes to input required user data:

- Choose folder – choose the folder containing the recordings to be classified
- Choose model – a fitted model to be used for classification
- Choose metadata – the file containing the fitted model metadata
- Time expanded – choose the correct time expansion factor, normally only used in recorders specifically intended for bat recordings. Can take the values "auto", 1 or 10. If the recording is not time expanded the value must be 1. If it's time expanded, the value 10 or "auto" can be selected. If "auto" is selected it is assumed that sampling rates < 50kHz corresponds to a value of 10 and sampling rates > 50kHz to corresponds to a value of 1
- Output file – the name of the files to store the results of the classification
- Irrelevant – does the fitted model includes an irrelevant class?
- Export plots – should a spectrogram of the classified recordings be saved to disk?

The classification results are stored in a folder called "output", created inside the folder containing the recordings. They are stored in a database in sqlite3 format with all the relevant events detected and the respective probability of belonging to a given class. Additionally a file in the csv format is saved to disk, containing summary statistics per recording, i.e. the class with most events detected in each particular recording and the average frequency of maximum energy of the events detected.

Usage

```
app_model()
```

Value

Starts the shiny app, no return value.

Author(s)

Bruno Silva

`auto_id` *Automatic classification of sound events in recordings*

Description

Run automatic classification of sound events on a set of recordings using a fitted model.

Usage

```
auto_id(model_path, update_progress = NA, metadata,
file_path, out_file, out_dir, save_png = TRUE, win_size = 50,
plot2console = FALSE, remove_noise = TRUE, recursive = FALSE, tx = 1)
```

Arguments

`model_path`	Character. Path to the fitted model.
`update_progress`	Progress bar only to be used inside shiny.
`metadata`	The object created with the function train_metadata() containing the parameters used to fit the model, or the path to the saved RDATA file.
`file_path`	Character. Path to the folder containing recordings to be classified by the fitted model.
`out_file`	Character. Name of the output file to save the results. Will be used to name the csv file and the sqlite database.
`out_dir`	Character. Path to the folder where the output results will be stored. Will be created if it doesn't exist already.
`save_png`	Logical. Should a spectrogram of the classified recordings with the identified event(s) and respective classification(s) be saved as png file?

win_size	Integer. Window size in ms to split recordings in chunks for classification. One peak per chunk is obtained and classified.
plot2console	Logical. Should a spectrogram of the classified recordings with the identified event(s) and respective classification(s) be plotted in the console while the analysis is running?
remove_noise	Logical. TRUE indicates that the model was fitted with a non-relevant class which will be deleted from the final output.
recursive	Logical. FALSE indicates that the recordings are in a single folder and TRUE indicates that there are recordings inside subfolders.
tx	Only used in recorders specifically intended for bat recordings. Can take the values "auto" or any numeric value. If the recording is not time expanded tx must be set to 1 (the default). If it's time expanded the numeric value corresponding to the time expansion should be indicated or "auto" should be selected. If tx = "auto" the function assumes that sampling rates < 50kHz corresponds to tx = 10 and > 50kHz to tx = 1.

Details

Runs a classification task on the recordings of a specified folder and saves the results of the analysis.

Value

Nothing.

Author(s)

Bruno Silva

---

create_db *Create a sqlite3 database*

Description

Create a sqlite3 database (if a database with the specified name doesn't exist already) with predefined tables. Two types of databases are possible, one to store recordings annotations and another to store the output of the classification.

Usage

```
create_db(path, db_name = NA, table_name = "labels",
type = "reference")
```

Arguments

path	Character. Path to the folder where the database will be created.
db_name	Character. Name of the database to be created.
table_name	Character. Name of the table to be created in the database. It is mandatory to use the default table name "labels" if the database is intended to be used in conjunction with other functions of this package.
type	Character indicating the type of database to create. Possible options are: "reference" which creates a database to be used to store recordings annotations for training purposes, and "id" which creates a database to output the results of the automatic classification.

Value

Nothing

Author(s)

Bruno Silva

Examples

```
## Not run:
dir_path <- tempdir()
create_db(dir_path,
db_name = "test",
table_name = "labels",
type = "reference")
file.remove(file.path(dir_path, "test.sqlite3"))

## End(Not run)
```

---

find_noise *Detect energy peaks in recordings with non-relevant events*

Description

Detects the temporal position of the desired number of energy peaks in a recording exclusively with non-relevant events.

Usage

```
find_noise(recording, nmax = 1, plot = FALSE)
```

Arguments

recording	Object of class "rc".
nmax	Integer indicating the maximum number of peaks to detect in the recording.
plot	Logical. If TRUE a plot showing the peak(s) is returned.

Value

A vector with the temporal position of the identified peak(s), in samples.

Author(s)

Bruno Silva

Examples

```
# Create a sample wav file in a temporary directory
recording <- tuneR::noise(duration = 44100)
temp_dir <- tempdir()
rec_path <- file.path(temp_dir, "recording.wav")
tuneR::writeWave(recording, filename = rec_path)
# Import the sample wav file
new_rec <- import_audio(rec_path, butt = FALSE, tx = 1)
find_noise(new_rec, nmax = 1, plot = FALSE)
file.remove(rec_path)
```

---

import_audio *Import a recording*

Description

Import a "wav" recording. If the recording is stereo it is converted to mono by keeping the channel with overall higher amplitude

Usage

```
import_audio(path, butt = TRUE, low, high, tx = 1)
```

Arguments

path	Character. Full path to the recording
butt	Logical. If TRUE filters the recording with a 12th order filter. The filter is applied twice to better cleaning of the recording
low	Minimum frequency in kHz for the butterworth filter
high	Maximum frequency in kHz for the butterworth filter
tx	Time expanded. Only used in recorders specifically intended for bat recordings. Can take the values "auto" or any numeric value. If the recording is not time expanded tx must be set to 1 (the default). If it's time expanded the numeric value corresponding to the time expansion should be indicated or "auto" should be selected. If tx = "auto" the function assumes that sampling rates < 50kHz corresponds to tx = 10 and > 50kHz to tx = 1.

Value

An object of class "rc". This object is a list with the following components:

- sound_samples – sound samples of the recording
- file_name – name of the recording
- file_time – time of modification of the file (indicated for Pettersson Elektronic detectors, for other manufactures creation time should be preferable but it's not implemented yet)
- fs – sample frequency
- tx – expanded time factor

Author(s)

Bruno Silva

Examples

```
# Create a sample wav file in a temporary directory
recording <- tuneR::sine(440)
temp_dir <- tempdir()
rec_path <- file.path(temp_dir, "recording.wav")
tuneR::writeWave(recording, filename = rec_path)
# Import the sample wav file
new_rec <- import_audio(rec_path, low = 1, high = 20, tx = 1)
new_rec
file.remove(rec_path)
```

---

ms2samples *Convert between time and number of samples in sound files*

Description

Convert time to number of samples or vice versa in sound files.

Usage

```
ms2samples(value, fs = 300000, tx = 1, inv = FALSE)
```

Arguments

value	Integer. Number of samples or time in ms.
fs	Integer. The sampling frequency in samples per second.
tx	Integer. Indicating the time expansion factor. If the recording is not time expanded tx must be set to 1 (the default).
inv	Logical. If TRUE converts time to number of samples, if FALSE number of samples to time.

Value

Integer. If inv = TRUE returns number of samples, if inv = FALSE returns time in ms.

Author(s)

Bruno Silva

Examples

```
ms2samples(150000, fs = 300000, tx = 1, inv = FALSE)
ms2samples(100, fs = 300000, tx = 1, inv = TRUE)
```

---

spectro_calls *Generate spectrograms from labels*

Description

Generate spectrograms from recording labels for classification purposes. The spectrogram parameters are user defined and should be selected depending on the type of sound event to classify.

Usage

```
spectro_calls(files_path, update_progress = NA,
db_path, spec_size = NA, window_length = NA,
frequency_resolution = 1, overlap = NA,
dynamic_range = NA, freq_range = NA, tx = 1, seed = 1002)
```

Arguments

files_path	Character. Path for the folder containing sound recordings.
update_progress	Progress bar only to be used inside shiny.
db_path	Character. Path for the database of recording labels created with the shinny app provided in the package.
spec_size	Integer. Spectrogram size in ms.
window_length	Numeric. Moving window length in ms.
frequency_resolution	Integer. Spectrogram frequency resolution with higher values meaning better resolution. Specifically, for any integer X provided, 1/X the analysis bandwidth (as determined by the number of samples in the analysis window) will be used. Not implemented yet, always uses 1 as input value.
overlap	Percentage of overlap between moving windows. Accepts values between 0.5 and 0.75.
dynamic_range	Threshold of minimum intensity values to show in the spectrogram. A value of 100 will typically be adequate for the majority of the recorders. If this is set to NULL, no threshold is applied.

freq_range	Frequency range of the spectrogram. Vector with two values, referring to the minimum and maximum frequency to show in the spectrogram.
tx	Time expanded. Only used in recorders specifically intended for bat recordings. Can take the values "auto" or any numeric value. If the recording is not time expanded tx must be set to 1 (the default). If it's time expanded the numeric value corresponding to the time expansion should be indicated or "auto" should be selected. If tx = "auto" the function assumes that sampling rates < 50kHz corresponds to tx = 10 and > 50kHz to tx = 1.
seed	Integer. Define a custom seed for randomizing data.

Value

A list with the following components:

- data_x – an array with the spectrogram matrices
- data_y – the labels for each matrix in one-hot-encoded format
- parameters – the parameters used to create the matrices
- labels_df – the labels with their respective numeric index

Author(s)

Bruno Silva

---

train_metadata *Obtain train metadata to run a fitted model*

Description

Obtain train metadata from the output of function spectro_calls. Needed to run a fitted model

Usage

```
train_metadata(train_data)
```

Arguments

train_data	Output of function spectro_calls.

Value

A list with the following components:

- parameters – parameters of the spectrograms
- classes – class names and respective codes

Author(s)

Bruno Silva

`%>%` *Pipe operator*

Description

See documentation of package magrittr for details.

Usage

```
lhs %>% rhs
```

Arguments

`lhs`	A value or the magrittr placeholder.
`rhs`	A function call using the magrittr semantics

Index

%>%, 12

app_label, 2
app_model, 3
auto_id, 5

create_db, 6

find_noise, 7

import_audio, 8

ms2samples, 9

spectro_calls, *4*, 10, *11*

train_metadata, 11