Package ‘soundClass’

January 22, 2022

Title Sound Classification Using Convolutional Neural Networks

Version 0.0.9

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 pipeline, 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 annotatedrecordings and fit CNN models
and 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.
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

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

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

Depends shinyBS, htmltools

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

app_label *Start app label recordings*

Description

Starts the app to label recordings

Usage

```
app_label()
```

app_model *Start app fit model*

Description

Starts the app to fit and run the model

Usage

```
app_model()
```

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 spectro_calls() containing the parameters used to fit the model.

`file_path` Character. Path to the folder containing recordings.

`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.

`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 using readWave and create a S3 object of class "rc". 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 = NA, time_step_size = 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. Note that this greatly impacts processing time, so adjust with care!

`time_step_size` Integer. Moving window step in ms.

`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

%>% *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