Package ‘soundClass’

August 17, 2021

Title Automatic Sound Classification with Convolutional Neural Networks

Version 0.0.0.9

Author Bruno Silva [aut, cre]

Maintainer Bruno Silva <bmsasilva@gmail.com>

Description All-in-one package for automatic classification of sound events using convolutional neural networks: manage and annotate large acoustic datasets in order to create and deploy automatic AI classifiers for sound events based on convolutional neural networks (CNN). It provides functionalities to annotate audio recordings and store information in database format, pre-process data for model training purposes and to deploy a full trained model to automatically classify sound events in new recordings.
By using automatic feature selection and a user-friendly GUI for managing data and traing/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.

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

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

Depends shinyBS, htmltools

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_shiny` *Automatic classification on a set of recordings*

Description

Applies automatic classification on a set of recordings using a fitted model

Usage

```
auto_id_shiny(
  model_path,
  updateProgress,
  metadata,
  file_path,
  out_file,
  out_dir,
  save_png = T,
  win_size = 50,
  plot2console = F,
  remove_noise = T,
  recursive = FALSE
)
```

Arguments

`model_path`	An object of class 'recording', created with the 'import_audio' function
`updateProgress`	Progress bar only to be used inside shiny
`metadata`	Parameters used to create train data for fitting the model
`file_path`	Path to the folder containing sound recordings
`out_file`	Character. Name of the file to output the results. Will be used to name the csv file and the sqlite database
`out_dir`	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`	Window size 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 ploted in the console while the analysis is running? as png file?
`remove_noise`	= T, 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

Details

Automatic classification on a set of recordings

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

Value

Nothing

Author(s)

Bruno Silva

---

butter_filter *Apply a buttterworth filter to sound samples*

Description

Apply a buttterworth filter, high pass or/and low pass, to sound samples. Based on the function butter

Usage

```
butter_filter(sound_samples, low = NA, high = NA, fs, tx, order = 12)
```

Arguments

`sound_samples`	Numeric vector with the sound samples to filter
`low`	Numeric. Minimum frequency in kHz for the butterworth filter
`high`	Numeric. Maximum frequency in kHz for the butterworth filter
`fs`	Integer with the sampling frequency of the recording
`tx`	Integer indicating the expanded time factor of the recording
`order`	Integer indicating the filter order to apply to the recording

Details

Buttterworth filter

Value

A vector with the filtered sound samples

Author(s)

Bruno Silva

Examples

```
sound <- runif(22000, min = -10000, max = 10000) # 1s sound sample
sound_filt <- butter_filter(sound,
  low = 4, high = 8,
  fs = 22000, tx = 1, order = 10
) # filter sound
```

`create_db` *Create a sqlite3 database*

Description

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

Usage

```
create_db(path, db_name, 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. inside the database. It is advisable to use the default table name "labels" if the databse 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

```
create_db(".//", db_name = "test", table_name = "labels",
type = "reference")
```

find_noise *Detect energy peaks in non-relevant recordings*

Description

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

Usage

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

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.

Details

Detect energy peaks in non-relevant recordings

Value

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

Author(s)

Bruno Silva

Examples

```
xxxxx
```

import_audio *Import a recording*

Description

Import a "wav" recording using readWave and create a S3 class object "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)
```

Arguments

`path` 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 correponding to the time expansion should be indicated or "auto" should be selected. If tx = "auto" the function assumes that sampling rates < 50kHz correponds to tx = 10 and > 50kHz to tx = 1.

Details

Import a recording

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

`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`	Numeric representing time in ms or number of samples.
`fs`	Integer. The sampling frequency in samples per second.
`tx`	Integer. Time expansion factor.
`inv`	Logical. If TRUE converts time to number of samples, if FALSE number of samples to time.

Details

Convert between time and number of samples in sound files

Value

if TRUE returns number of samples, if 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` *Generates spectrograms from recording labels.*

Description

Generates spectrograms from recording's 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, updateProgress,
db_path, spec_size = NA, window_length = NA,
frequency_resolution = NA, time_step_size = NA, dynamic_range = NA,
freq_range = NA)
```

Arguments

`files_path`	Path for the folder containing sound recordings
`updateProgress`	Progress bar only to be used inside shiny
`db_path`	Path for the database of recording labels created with the shinny app provided in the package

`spec_size` Spectrogram size in ms

`window_length`
Moving window length to create the spectrogram in ms

`frequency_resolution`
Spectrogram frequency resolution with higher numbers 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`
Moving window step in ms

`dynamic_range`
Threshold of minimum intensity values to show in the spectrogram

`freq_range` Frequency range of the spectrogram. Vector with two values, refering to the minimum and maximum frequency to show in the spectrogram

Details

Generate spectrograms from labels

Value

A list with the spectrogram and the respective label

Author(s)

Bruno Silva

`train_data_process` *Process train data for Keras format*

Description

Processes the train data outputed by function 'spec_calls' to Keras format.

Usage

```
train_data_process(rdata_list, seed = 1002)
```

Arguments

`rdata_list` An object created with the 'spec_calls' function.

`seed` Integer. Used for the randomization of the observations.

Details

Process train data for Keras format

The spectrogram matrices are converted into an array (XX train data) and the labels one-hot-encoded (YY train data). The observations are also randomized.

Value

A list with two components: data_x, an array with the XX data and data_y, the labels one-hot-encoded.#’

Author(s)

Bruno Silva

`%>%` *Pipe operator*

Description

See `magrittr::%>%` for details.

Usage

```
lhs %>% rhs
```

Arguments

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