BirdVoxClassify Documentation

Release 0.3.2

Jason Cramer, Vincent Lostanlen, Justin Salamon, Andrew Farnsw

Jul 06, 2021

Contents

CHAPTER 1

Installation instructions

1.1 Dependencies

1.1.1 Python Versions

Currently, we support Python 3.6 and Python 3.7. Python 3.8 is not currently supported since TensorFlow 1.x is not supported for Python 3.8.

1.1.2 Python Versions

Currently, we support Python 3.6, 3.7, and 3.8.

1.1.3 libsndfile (Linux only)

BirdVoxClassify depends on the PySoundFile module to load audio files, which itself depends on the non-Python library libsndfile. On Windows and Mac OS X, these will be installed automatically via the `pip` package manager and you can therefore skip this step. However, on Linux, `libsndfile` must be installed manually via your platform's package manager. For Debian-based distributions (such as Ubuntu), this can be done by simply running

```
apt-get install libsndfile
```

For more detailed information, please consult the installation instructions of pysoundfile.

1.1.4 Note about TensorFlow:

We have dropped support for Tensorflow 1.x, and have moved to Tensorflow 2.x.

1.2 Installing BirdVoxClassify

The simplest way to install BirdVoxClassify is by using `pip`, which will also install the additional required dependencies if needed. To install the latest stable version of BirdVoxClassify using `pip`, simply run Oh, Pyth

```
pip install birdvoxclassify
```

To install the latest version of BirdVoxClassify from source:

1. Clone or pull the latest version:

   ```
   git clone git@github.com:BirdVox/birdvoxclassify.git
   ```

2. Install using pip to handle Python dependencies:

   ```
   cd birdvoxclassify
   pip install -e .
   ```

CHAPTER 2

BirdVoxClassify tutorial

2.1 Introduction

Welcome to the BirdVoxClassify tutorial! In this tutorial, we'll show how to use BirdVoxClassify to classify the species of bird flight calls in audio clips . The supported audio formats are those supported by the *pysoundfile* library, which is used for loading the audio (e.g. WAV, OGG, FLAC).

2.2 Using the Library

You can simply compute bird species predictions out of the box, like so:

```
import birdvoxclassify as bvc
import json
filepath = '/path/to/file.wav'
filepath_list = [
    '/path/to/file1.wav',
    '/path/to/file2.wav',
    '/path/to/file3.wav'
]
## Get prediction dictionary object
# Prediction for a single file
formatted_pred = process_file(filepath)
# Prediction for a list of files
formatted_pred = process_file(filepath_list)
# Dictionary of best predicted candidates for a list of files. Hierarchical␣
↪consistency is applied by default.
best_candidates = process_file(filepath_list,
                               select_best_candidates=True)
# Get candidates without applying hierarchical consistency
best_candidates = process_file(filepath_list,
                               select_best_candidates=True,
                               hierarchical_consistency=False)
```

(continues on next page)

(continued from previous page)

```
## Save individual output files for each audio file
process_file(filepath_list, output_dir='/output/dir')
# Add a suffix to output filenames (e.g. /path/to/file1_<suffix>.json)
process_file(filepath_list, output_dir='/output/dir', suffix='suffix')

# Save a summary output file
process_file(filepath_list, output_summary_path='/path/to/output/file.json')

# Specify model (and taxonomy) to use
formatted_pred = process_file(filepath, model_name=bvc.DEFAULT_MODEL_NAME)

# Pre-load model and taxonomy
model = bvc.load_classifier(bvc.DEFAULT_MODEL_NAME)
taxonomy_path = bvc.get_taxonomy_path(bvc.DEFAULT_MODEL_NAME)
taxonomy = bvc.load_taxonomy(taxonomy_path)
formatted_pred = process_file(filepath, classifier=model, taxonomy=taxonomy)

# Change batch size depending on computational resources
formatted_pred = process_file(filepath, batch_size=32)
```

You can also compute predictions directly on loaded audio arrays:

```
import birdvoxclassify as bvc
import soundfile as sf
import json

# Load audio
audio, sr = sf.read('/path/to/file.wav')
pcen = bvc.compute_pcen(audio, sr, input_format=True)
# Load model and taxonomy
model = bvc.load_classifier(bvc.DEFAULT_MODEL_NAME)
taxonomy_path = bvc.get_taxonomy_path(bvc.DEFAULT_MODEL_NAME)
# IMPORTANT: Use this utility instead of loading it manually to ensure
#            proper behavior
taxonomy = bvc.load_taxonomy(taxonomy_path)

# Get list of one-hot prediction array for each level of the taxonomy
pred_list = bvc.predict(pcen, model)
coarse_pred, medium_pred, fine_pred = pred_list

# Format prediction in more interpretable format
formatted_pred = bvc.format_pred(pred_list, taxonomy)

# Select best candidates from prediction. Hierarchical consistency is applied by␣
↪default.
best_candidates = get_best_candidates(pred_list=pred_list, taxonomy=taxonomy)
# Can use prediction list or formatted prediction dict.
best_candidates = get_best_candidates(formatted_pred_dict=formatted_pred,
                                      taxonomy=taxonomy)
# Get candidates without applying hierarchical consistency
best_candidates = get_best_candidates(formatted_pred_dict=formatted_pred,
                                      hierarchical_consistency=False)
```

2.3 Using the Command Line Interface (CLI)

To generate predictions for a single file via the command line run:

```
$ birdvoxclassify /path/to/file.wav
```

This will print out the model prediction in JSON format. If you wish, you can output only the best candidates (at each taxonomic level):

```
$ birdvoxclassify -B /path/to/file.wav
```

This will print out the best candidates under the model prediction in JSON format. BirdVoxClassify applies hierarchical consistency to the candidates, but it can be disabled as follows:

```
$ birdvoxclassify -B -N /path/to/file.wav
```

You can also provide multiple input files or directories:

```
$ birdvoxclassify /path/to/file1.wav /path/to/file2.wav /path/to/file3.wav
```

You can set the output directory for per-file output files as follows:

```
$ birdvoxclassify /path/to/file1.wav /path/to/file2.wav /path/to/file3.wav --output-
↪dir /output/dir
```

This will create an output files `/output/dir/file1.json`, `/output/dir/file2.json`, and `/output/dir/file3.json`.

You can create a single summary output file as follows:

```
$ birdvoxclassify /path/to/file1.wav /path/to/file2.wav /path/to/file3.wav --output-
↪summary-path /output/summary/path.json
```

which will create a summary output file at `/output/summary/path.json`.

You can specify the classifier model name as follows:

```
$ birdvoxclassify /path/to/file.wav --classifier-name birdvoxclassify-flat-multitask-
↪convnet_tv1hierarchical-3c6d869456b2705ea5805b6b7d08f870
```

If processing a large number of files, you can set the prediction batch size appropriately for your computational resources as follows:

```
$ birdvoxclassify /large/audio/dir --batch-size 128
```

You can append a suffix to the output files as follows:

```
$ birdvoxclassify /path/to/file1.wav /path/to/file2.wav /path/to/file3.wav --output-
↪dir /output/dir --suffix suffix
```

This will create an output files `/output/dir/file1_suffix.json`, `/output/dir/file2_suffix.json`, and `/output/dir/file3_suffix.json`.

You can print verbose outputs by running:

```
$ birdvoxclassify /path/to/file.wav --verbose
```

Finally, you can suppress non-error printouts by running:

```
$ birdvoxclassify /path/to/file.wav --quiet
```

CHAPTER 3

BirdVoxClassify taxonomy

3.1 Introduction

In the context of the BirdVox project, we limit the taxonomical scope to a subset of bird species. This subset is formalized via a taxonomy file which is used to both specify what species we are interested in as well as describing the output of a particular model using the taxonomy. In fact, model names refer to the taxonomy they use (as well as the MD5 checksum of the taxonomy file) in their filenames. For example, the model with ID `birdvoxclassify-flat-multitask-convnet_tv1hierarchical-3c6d869456b2705ea5805b6b7d08f870` uses the taxonomy `tv1hierarchical`, which has an MD5 checksum of `3c6d869456b2705ea5805b6b7d08f870`. Generally, a model name is in the format `<model identifier>_<taxonomy name>-<taxonomy md5 checksum>`.

v0.3.1 UPDATE: model names with taxonomy md5sum ''2e7e1bbd434a35b3961e315cfe3832fc'' or ''beb9234f0e13a34c7ac41db72e85addd'' are not available in this version but are restored in v0.3.1 for backwards compatibility. They will no longer be supported starting with v0.4. Please use model names with taxonomy md5 checksums ''3c6d869456b2705ea5805b6b7d08f870'' and ''2f6efd9017669ef5198e48d8ec7dce4c'' (respectively) instead.

3.2 Overview

The primary taxonomy currently used (`v1hierarchical`) is as such:

- [0] Other - Coarse (Non-passerine)
- **[1] Passerines**
  - [1.0] Other - Medium (Other Passerine)
  - **[1.1] American Sparrow**
    - [1.1.0] Other - Fine (Other American Sparrow)
    - [1.1.1] ATSP (American Tree Sparrow)
    - [1.1.2] CHSP (Chipping Sparrow)

        - [1.1.3] SAVS (Savannah Sparrow)
        - [1.1.4] WTSP (White-throated Sparrow)
        - [1.1.X] Unknown - Fine (Unknown American Sparrow)
    - **[1.2] Cardinal**
        - [1.2.0] Other - Fine (Other Cardinal)
        - [1.2.1] RBGR (Rose-breasted Grosbeak)
        - [1.2.X] Unknown - Fine (Unknown Cardinal)
    - **[1.3] Thrush**
        - [1.3.0] Other - Fine (Other Thrush)
        - [1.3.1] GCTH (Gray-cheeked Thrush)
        - [1.3.2] SWTH (Swainson's Thrush)
        - [1.3.X] Unknown - Fine (Unknown Thrush)
    - **[1.4] Warbler**
        - [1.4.0] Other - Fine (Other Warbler)
        - [1.4.1] AMRE (American Redstart)
        - [1.4.2] BBWA (Bay-breasted Warbler)
        - [1.4.3] BTBW (Black-throated Blue Warbler)
        - [1.4.4] CAWA (Canada Warbler)
        - [1.4.5] COYE (Common Yellowthroat)
        - [1.4.6] MOWA (Mourning Warbler)
        - [1.4.7] OVEN (Ovenbird)
        - [1.4.X] Unknown - Fine (Unknown Warbler)
    - [1.X] Unknown - Medium (Unknown Passerine)
- [X] Unknown - Coarse (Unknown bird)

Note that all species have a fully qualified three-digit taxonomy code. Each digit corresponds to the respective values of each species within our three level taxonomy, corresponding roughly to the standard taxonomical groupings "order", "family", and "species". N-digit taxonomy codes (e.g. for N < 3) correspond to higher level concepts in the taxonomy (such as family). Any N-digit taxonomy code ending in a `0` corresponds to an "other" class, simply meaning it is outside our scope of interest. For example, `1.0` corresponds to an "other Passerine" meaning a Passerine that is not a member of the families of interest. Similarly, any N-digit taxonomy code ending in an `X` corresponds to an "unknown" class, meaning that the annotator could not determine the identity of a particular bird at that level of the taxonomy. For example, `1.1.X` corresponds to an American Sparrow for which the annotator was unsure of the particular species.

3.3 Taxonomy format

The taxonomy is specified in JSON format. The taxonomy files can be found in `<BirdVoxClassify dir>/resources/taxonomy/`.

```
{
  "taxonomy": [
    {
      "id": <str>, // Numeric taxonomic ID
      "common_name": <str>, // Common name
      "scientific_name": <str>, // Latin name
      "taxonomy_level_names": <str>, // Level of taxonomy
      "taxonomy_level_aliases": {

      },
      "child_ids": [
          <str>, // Child taxonomic IDs
          ...
      ]
    },
    ...

  ],
  "output_encoding": {
    <str>: [ // Name of level of taxonomy
      // Ordered list of outputs
      {
        "ids": [
            <str>, // List of taxonomic IDs encapsulated in this output
            ...
        ]
      },
      ...
      {
        "ids": [
            <str>, // Last output should encompass all "other" classes
            ...
        ]
      }
    ],
}
```

The `taxonomy` field contains nodes of the tree of the taxonomy. each of which contain the N-digit taxonomy reference ID, identifying information and aliases about the node, and the IDs of children nodes in the taxonomy. `output_encoding` specifies the taxonomy IDs associated with each element of an output probability vector produced by a classifier using this taxonomy. The order of the list associated with each level of the taxonomy corresponds to the position in an output vector.

3.4 Output format

Model output is given as JSON:

```
{
  <prediction level> : {
    <taxonomy id> : {
      "probability": <float>,
      "common_name": <str>,
      "scientific_name": <str>,
      "taxonomy_level_names": <str>,
      "taxonomy_level_aliases": <dict of aliases>,
```

(continues on next page)

(continued from previous page)

```
      "child_ids": []
    },
    ...
    "other" : {
      "common_name": "other",
      "scientific_name": "other",
      "taxonomy_level_names": level,
      "taxonomy_level_aliases": {},
      "child_ids": <list of children IDs>
    }
  },
  ...
}
```

The probabilities at each prediction level. For a summary file, containing predictions for multiple files the output is given as:

```
{
  <filename> : {
    <output node>
  },
  ...
}
```

BirdVoxClassify is an open-source Python library for classifying the species contained in short bird flight call recordings.

CHAPTER 4

API Reference

4.1 API Reference

4.1.1 Core functionality

birdvoxclassify.core.**apply_hierarchical_consistency**(*formatted_pred_dict*, *taxonomy*, *level_threshold_dict=None*, *detection_threshold=0.5*)

Obtain the best predicted candidate class for a prediction at all taxonomic levels, enforcing "top-down" hierarchical consistency. That is, starting from the "coarsest" taxonomic level, if the most probable class is considered "present" (estimated probability greater than a threshold), it is considered the best candidate for that level, and only taxonomic children of this class will be considered when choosing candidates for "finer" taxonomic levels. If the most probable class is not considered "present" (estimated probability below the same threshold), then the "other" class is chosen as the best candidate, with the probability assigned to be the complement of the most probable "consistent" class.

Parameters

formatted_pred_dict [dict] Formatted dictionary of predictions.

taxonomy [dict or None [default: `None`]] Taxonomy JSON object used to apply hierarchical consistency. If `None`, then `hierarchical_consistency` must be `False`.

level_threshold_dict [dict or None [default: `None`]] Optional dictionary of detection thresholds for each taxonomic level.

detection_threshold [float [default: `0.5`]] Detection threshold applied uniformly to all classes at all levels. If `level_threshold_dict` is provided, this is ignored.

Returns

best_candidates_dict [dict] Formatted dictionary specifying the best candidate for each taxonomic level.

birdvoxclassify.core.**batch_generator**(*filepath_list*, *batch_size=512*)

Returns a generator that, from a list of filepaths, yields batches of PCEN images and the corresponding filenames.

Parameters

filepath_list [list[str]] (Non-empty) list of filepaths to audio files for which to generate batches of PCEN images and the corresponding filenames

batch_size [int [default: 512]] Size of yielded batches

Yields

batch [np.ndarray [shape: (batch_size, top_freq_id, n_hops, 1)]] PCEN batch

batch_filepaths [list[str]] List of filepaths corresponding to the clips in the batch

birdvoxclassify.core.**compute_pcen**(*audio*, *sr*, *input_format=True*)

Computes PCEN (per-channel-energy normalization) for the given audio clip.

Parameters

audio [np.ndarray [shape: (N,)]] Audio array

sr [int] Sample rate

input_format [bool [default: True]] If True, adds an additional channel dimension (of size 1) and ensures that a fixed number of PCEN frames (corresponding to `get_pcen_settings()['n_hops']`) is returned. If number of frames is greater, the center frames are returned. If the the number of frames is less, empty frames are padded.

Returns

pcen [np.ndarray [shape: (top_freq_id, n_hops, 1) or (top_freq_id, num_frames)]] Per-channel energy normalization processed Mel spectrogram. If `input_format=True`, will be in shape `(top_freq_id, n_hops, 1)`. Otherwise it will be in shape `(top_freq_id, num_frames)`, where `num_frames` is the number of PCEN frames for the entire audio clip.

birdvoxclassify.core.**format_pred**(*pred_list*, *taxonomy*)

Formats a list of predictions for a single audio clip into a more human-readable JSON object using the given taxonomy object.

The output will be in the following format:

```
{
  <prediction level> : {
    <taxonomy id> : {
      "probability": <float>,
      "common_name": <str>,
      "scientific_name": <str>,
      "taxonomy_level_names": <str>,
      "taxonomy_level_aliases": <dict of aliases>,
      "child_ids": <list of children IDs>
    },
    ...
  },
  ...
}
```

Parameters

pred_list [list[np.ndarray [shape (1, num_labels) or (num_labels,)]] List of predictions at the taxonomical levels predicted by the model for a single example. `num_labels` may be different for each of the different levels of the taxonomy.

taxonomy [dict] Taxonomy JSON object

Returns

formatted_pred_dict [dict] Prediction dictionary object

`birdvoxclassify.core.`**`format_pred_batch`**(*batch_pred_list*, *taxonomy*)

Formats a list of predictions for a batch of audio clips into a more human-readable JSON object using the given taxonomy object. The output will be in the form of a list of JSON objects in the format returned by `format_pred`.

Parameters

batch_pred_list [list[np.ndarray [shape (batch_size, num_labels)]]] List of predictions at the taxonomical levels predicted by the model for a batch of examples. `num_labels` may be different for each of the different levels of the taxonomy.

taxonomy [dict] Taxonomy JSON object

Returns

pred_dict_list [list[dict]] List of JSON dictionary objects

`birdvoxclassify.core.`**`get_batch_best_candidates`**(*batch_pred_list=None*, *batch_formatted_pred_list=None*, *taxonomy=None*, *hierarchical_consistency=True*)

Obtain the best candidate classes for each prediction in a batch.

Parameters

batch_pred_list [list or None [default: `None`]] List of batch predictions. If not provided, `batch_formatted_pred_list` must be provided.

batch_formatted_pred_list [list or None [default: `None`]] List of formatted batch predictions. If not provided, `batch_pred_list` must be provided.

taxonomy [dict or None [default: `None`]] Taxonomy JSON object used to apply hierarchical consistency. If `None`, then `hierarchical_consistency` must be `False`.

hierarchical_consistency [bool [default: `True`]] If `True`, apply hierarchical consistency to predictions.

Returns

batch_best_candidates_list [list] List of formatted dictionaries specifying the best candidates for each taxonomic level.

`birdvoxclassify.core.`**`get_best_candidates`**(*pred_list=None*, *formatted_pred_dict=None*, *taxonomy=None*, *hierarchical_consistency=True*)

Obtain the best predicted candidate class for a prediction at all taxonomic levels. The output will be in the following format:

```
{
  <prediction level> : {
    "probability": <float>,
    "common_name": <str>,
    "scientific_name": <str>,
    "taxonomy_level_names": <str>,
```

(continues on next page)

(continued from previous page)

```
    "taxonomy_level_aliases": <dict of aliases>,
    "child_ids": <list of children IDs>
  },
  ...
}
```

Parameters

pred_list [list[np.ndarray [shape (1, num_labels) or (num_labels,)] or None [default: `None`]] List of predictions at the taxonomical levels predicted by the model for a single example. If provided, `taxonomy`, must also be provided.

If not provided, `formatted_pred_dict` must be provided.

formatted_pred_dict [dict or None [default: `None`]] Formatted dictionary of predictions. If not provided, `pred_list` must be provided.

taxonomy [dict or None [default: `None`]] Taxonomy JSON object used to apply hierarchical consistency. If `None`, then `hierarchical_consistency` must be `False`.

hierarchical_consistency [bool [default: `True`]] If `True`, apply hierarchical consistency to predictions.

Returns

best_candidates_dict [dict] Formatted dictionary specifying the best candidate for each taxonomic level.

`birdvoxclassify.core.`**`get_model_path`**(*model_name*)

Returns path to the bird species classification model of the given name.

Parameters

model_name [str] Name of classifier model. Should be in format `<model id>_<taxonomy version>-<taxonomy md5sum>`. *v0.3.1 UPDATE: model names with taxonomy md5 checksum 2e7e1bbd434a35b3961e315cfe3832fc or beb9234f0e13a34c7ac41db72e85addd are not available in this version but are restored in v0.3.1 for backwards compatibility. They will no longer be supported starting with v0.4. Please use model names with taxonomy md5 checksums 3c6d869456b2705ea5805b6b7d08f870 and 2f6efd9017669ef5198e48d8ec7dce4c (respectively) instead.*

Returns

model_path [str] Path to classifier model weights. Should be in format `<BirdVoxClassify dir>/resources/models/<model id>_<taxonomy version>-<taxonomy md5sum>.h5`

`birdvoxclassify.core.`**`get_output_path`**(*filepath*, *suffix*, *output_dir*)

Returns output path to file containing bird species classification predictions for a given audio clip file.

Parameters

filepath [str] Path to audio file to be processed

suffix [str] String to append to filename (including extension)

output_dir [str or None] Path to directory where file will be saved. If None, will use directory of given filepath.

Returns

output_path [str] Path to output file

birdvoxclassify.core.**get_pcen_settings**()

Returns dictionary of Mel spectrogram and PCEN parameters for preparing the input to the bird species classification models.

Returns

pcen_settings [dict[str, *]] Dictionary of Mel spectrogram and PCEN parameters

birdvoxclassify.core.**get_taxonomy_node**(*ref_id*, *taxonomy*)

Gets node in taxonomy corresponding to the given reference ID (e.g. `1.4.1`)

Parameters

ref_id [str] Taxonomy reference ID

taxonomy [dict] Taxonomy JSON object

Returns

node [dict[str, *]] Taxonomy node, containing information about the entity corresponding to the given taxonomy reference ID

birdvoxclassify.core.**get_taxonomy_path**(*model_name*)

Get the path to the taxonomy corresponding to the model of the given name.

Specifically, with a model name of the format:

`<model id>_<taxonomy version>-<taxonomy md5sum>`

the path to taxonomy file `<BirdVoxClassify dir>/resources/taxonomy/<taxonomy version>.json` is returned. The MD5 checksum of this file is compared to <taxonomy md5sum> to ensure that the content of the taxonomy file matches the format of the output that the model is expected to produce.

Parameters

model_name [str] Name of model. Should be in format *<model id>_<taxonomy version>-<taxonomy md5sum>*. *v0.3.1 UPDATE: model names with taxonomy md5 checksums 2e7e1bbd434a35b3961e315cfe3832fc or beb9234f0e13a34c7ac41db72e85addd are not available in this version but are restored in v0.3.1 for backwards compatibility. They will no longer be supported starting with v0.4. Please use model names with taxonomy md5 checksums 3c6d869456b2705ea5805b6b7d08f870 and 2f6efd9017669ef5198e48d8ec7dce4c (respectively) instead.*

Returns

taxonomy_path [str] Path to taxonomy file, which should be in format *<BirdVoxClassify dir>/resources/taxonomy/<taxonomy version>.json*

birdvoxclassify.core.**load_classifier**(*model_name*)

Loads bird species classification model of the given name.

Parameters

model_name [str] Name of classifier model. Should be in format `<model id>_<taxonomy version>-<taxonomy md5sum>`. *v0.3.1 UPDATE: model names with taxonomy md5 checksum 2e7e1bbd434a35b3961e315cfe3832fc or beb9234f0e13a34c7ac41db72e85addd are not available in this version but are restored in v0.3.1 for backwards compatibility. They will no longer be supported starting with v0.4. Please use model names with taxonomy md5 checksums 3c6d869456b2705ea5805b6b7d08f870 and 2f6efd9017669ef5198e48d8ec7dce4c (respectively) instead.*

Returns

classifier [keras.models.Model] Bird species classification model

birdvoxclassify.core.**load_taxonomy**(*taxonomy_path*)

Loads taxonomy JSON file as an OrderedDict to ensure consistent ordering. Taxonomy files specify output encodings in order from coarse to fine by convention.

Please use this function instead of manually loading the taxonomy!

Parameters

taxonomy_path [str] Path to taxonomy file.

Returns

taxonomy [OrderedDict] Taxonomy object

birdvoxclassify.core.**predict**(*pcen*, *classifier*, *logger_level=20*)

Performs bird species classification on PCEN arrays using the given model.

Parameters

pcen [np.ndarray [shape (n_mels, n_hops, 1) or (batch_size, n_mels, n_hops, 1)] PCEN array for a single clip or a batch of clips

classifier [keras.models.Model] Bird species classification model object

logger_level [int [default: `logging.INFO`]] Logger level

Returns

pred_list [list[np.ndarray [shape (batch_size or 1, num_labels)]]] List of predictions at the taxonomical levels predicted by the model. num_labels may be different for each of the different levels of the taxonomy. If a single example is given (i.e. there is no batch dimension in the input PCEN), `batch_size = 1`.

birdvoxclassify.core.**process_file**(*filepaths*, *output_dir=None*, *output_summary_path=None*, *classifier=None*, *taxonomy=None*, *batch_size=512*, *suffix=''*, *select_best_candidates=False*, *hierarchical_consistency=True*, *logger_level=20*, *model_name='birdvoxclassify-taxonet_tv1hierarchical-3c6d869456b2705ea5805b6b7d08f870'*)

Runs bird species classification model on one or more audio clips.

Parameters

filepaths [list or str] Filepath or list of filepaths of audio files for which to run prediction

output_dir [str or None [default: `None`]] Output directory used for outputting per-file prediction JSON files. If `None`, no per-file prediction JSON files are produced.

output_summary_path [str or None [default: `None`]] Output path for summary prediction JSON file for all processed audio files. If `None`, no summary prediction file is produced.

classifier [keras.models.Model or None [default: `None`]] Bird species classification model object. If `None`, the model corresponding to `model_name` is loaded.

taxonomy [dict or None [default: `None`]] Taxonomy JSON object. If `None`, the taxonomy corresponding to `model_name` is loaded.

batch_size [int [default: `512`]] Batch size for predictions

suffix [str [default: `""`]] String to append to filename

select_best_candidates [bool [default: `False`]] If `True`, best candidates will be provided in output dictionary instead of all classes and their probabilities.

hierarchical_consistency [bool [default: `True`]] If `True` and if `select_best_candidates` is `True`, apply hierarchical consistency when selecting best candidates.

logger_level [int [default: `logging.INFO`]] Logger level

model_name [str [default birdvoxclassify.DEFAULT_MODEL_NAME]] Name of classifier model. Should be in format `<model id>_<taxonomy version>-<taxonomy md5sum>`. *v0.3.1 UPDATE: model names with taxonomy md5sum 2e7e1bbd434a35b3961e315cfe3832fc or beb9234f0e13a34c7ac41db72e85addd are not available in this version but are restored in v0.3.1 for backwards compatibility. They will no longer be supported starting with v0.4. Please use model names with taxonomy md5 checksums 3c6d869456b2705ea5805b6b7d08f870 and 2f6efd9017669ef5198e48d8ec7dce4c (respectively) instead.*

Returns

output_dict [dict[str, dict]] Output dictionary mapping audio filename to prediction dictionary. If `select_best_candidates` is `False`, the dictionary is in the format produced by `format_pred`. Otherwise, the dictionary is in the format produced by `get_best_candidates`.

CHAPTER 5

Contribute

- Issue tracker
- Source code

CHAPTER 6

Changes

6.1 Changelog

6.1.1 v0.3.1

- Restore deprecated taxonomy files
- Raise DeprecationWarning when deprecated taxonomy files or models that use them are loaded
- Update documentation to notify users of deprecation of these files and how to address it
- Update v0.3.0 changelog

6.1.2 v0.3.0

- Add functionality for obtaining best candidates from predictions
- Add hierarchical consistency implementation for selecting best candidates
- Drop `six` dependency.
- Update taxonomy files so that order-level taxa are in plural form *[v0.3.1 UPDATE: model names with taxonomy md5sum ‘‘2e7e1bbd434a35b3961e315cfe3832fc‘‘ or ‘‘beb9234f0e13a34c7ac41db72e85addd‘‘ are not available in this version but are restored in v0.3.1 for backwards compatibility. They will no longer be supported starting with v0.4. Please use model names with taxonomy md5 checksums ‘‘3c6d869456b2705ea5805b6b7d08f870‘‘ and ‘‘2f6efd9017669ef5198e48d8ec7dce4c‘‘ (respectively) instead.]*

6.1.3 v0.2.0

- Drop support for Python 3.5, add support for Python 3.7 and 3.8.
- Deprecate and remove v1 model non-hierarchical
- Add models compatible with Python 3.8

- Make TaxoNet the default model.
- Fix broken dependencies.
- Swap `keras` for `tf.keras` and require Tensorflow 2.x.

6.1.4 v0.1.1

- Add TaxoNet model.

6.1.5 v0.1.0

- First release.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

Index