edgel3

Release 0.2.1

Apr 19, 2021

Contents

CHAPTER 1

Installation instructions

1.1 Dependencies

1.1.1 Tensorflow

`edgel3` has been tested with Tensorflow 2.0 and Keras 2.3.1.

```
>>> pip install tensorflow==2.0.0
```

1.1.2 libsndfile

`edgel3` depends on the *pysoundfile* module to load audio files, which depends on the non-Python library `libsndfile`. On Windows and macOS, these will be installed via `pip` and you can therefore skip this step. However, on Linux this 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 libsndfile1
```

For more detailed information, please consult the pysoundfile installation documentation.

1.2 Installing edgel3

The simplest way to install `edgel3` is by using `pip`, which will also install the additional required dependencies if needed. To install `edgel3` using `pip`, simply run

```
>>> pip install edgel3
```

To install the latest version of `edgel3` from source:

1. Clone or pull the lastest version:

```
>>> git clone https://github.com/ksangeeta2429/edgel3.git
```

2. Install using pip to handle python dependencies:

```
>>> cd edgel3
>>> pip install -e .
```

CHAPTER 2

Tutorial

2.1 Introduction

With **edgel3**, you can compute audio embeddings from smaller versions of L3 models that can be useful for resource constrained devices. 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

edgel3 supports two types of `model_type`:

- `sparse`: sparse L3 audio
- `sea`: SONYC-UST specialized L3 audio

The deafult audio model is 95.45% pruned and fine-tuned sparse L3 audio. You can compute audio embeddings out of default model by:

```
import edgel3
import soundfile as sf

audio, sr = sf.read('/path/to/file.wav')
emb, ts = edgel3.get_embedding(audio, sr)
```

`get_embedding` returns two objects. The first object `emb` is a T-by-D numpy array, where T is the number of analysis frames used to compute embeddings, and D is the dimensionality of the embedding. The second object `ts` is a length-T numpy array containing timestamps corresponding to each embedding (to the center of the analysis window, by default).

These defaults for **sparse** models be changed via the following optional parameters:

- `sparsity`: 53.5, 63.5, 72.3, 87.0, or 95.45 (default)
- `retrain_type`: “kd”, “ft” (default)

For example, to get embedding out of 81.0% sparse audio model that has been trained with knowledge-distillation method, you can use:

```
import edgel3
import soundfile as sf

audio, sr = sf.read('/path/to/file.wav')
emb, ts = edgel3.get_embedding(audio, sr, model_type='sparse', retrain_type='kd',␣
↪sparsity=81.0)
```

All **sea** models have reduced input representation. Moreover, models with embedding dimension < 512 also have reduced architecture. The default embedding dimension for `sea` models is 128 and it can be changed with the help of `emb_dim` parameter:

- `emb_dim`: 512, 256, 128 (default), 64

```
import edgel3
import soundfile as sf

audio, sr = sf.read('/path/to/file.wav')
emb, ts = edgel3.get_embedding(audio, sr, model_type='sea', emb_dim=256)
```

By default edgel3 will pad the beginning of the input audio signal by 0.5 seconds (half of the window size) so that the the center of the first window corresponds to the beginning of the signal ("zero centered"), and the returned timestamps correspond to the center of each window. You can disable this centering like this:

```
emb, ts = edgel3.get_embedding(audio, sr, center=True)
```

The hop size used to extract the embedding is 0.1 seconds by default (i.e. an embedding frame rate of 10 Hz). In the following example we change the hop size from 0.1 (10 frames per second) to 0.5 (2 frames per second):

```
emb, ts = edgel3.get_embedding(audio, sr, hop_size=0.5)
```

Finally, you can silence the Keras printout during inference (verbosity) by changing it from 1 (default) to 0:

```
emb, ts = edgel3.get_embedding(audio, sr, verbose=0)
```

By default, the model file is loaded from disk every time `get_embedding` is called. To avoid unnecessary I/O when processing multiple files with the same model, you can load it manually and pass it to the function via the `model` parameter:

```
model = edgel3.models.load_embedding_model(model_type='sparse', retrain_type='ft',␣
↪sparsity=53.5)
emb1, ts1 = edgel3.get_embedding(audio1, sr1, model=model)
emb2, ts2 = edgel3.get_embedding(audio2, sr2, model=model)
```

Since the model is provided, keyword arguments `model_type` and all parameters associated with `sea` and `sparse` will be ignored.

To compute embeddings for an audio file from a given model and save them to the disk, you can use `process_file`:

```
import edgel3
import numpy as np

audio_filepath = '/path/to/file.wav'

# Save the embedding output to '/path/to/file.npz'
edgel3.process_file(audio_filepath)
```

(continues on next page)

(continued from previous page)

```
# Saves the embedding output to '/path/to/file_suffix.npz'
edgel3.process_file(audio_filepath, suffix='suffix')

# Saves the embedding output to `/different/dir/file_suffix.npz`
edgel3.process_file(audio_filepath, output_dir='/different/dir', suffix='suffix')
```

The embddings can be loaded from disk using numpy:

```
import numpy as np

data = np.load('/path/to/file.npz')
emb, ts = data['embedding'], data['timestamps']
```

As with `get_embedding`, you can load the model manually and pass it to `process_file` to avoid loading the model multiple times:

```
import edgel3
import numpy as np

model = edgel3.models.load_embedding_model(model_type='sparse', retrain_type='ft',␣
↪sparsity=53.5)

audio_filepath = '/path/to/file.wav'

# Save the embedding output to '/path/to/file.npz'
edgel3.process_file(audio_filepath, model=model)

# Saves the embedding output to '/path/to/file_suffix.npz'
edgel3.process_file(audio_filepath, model=model, suffix='suffix')

# Saves the embedding output to `/different/dir/file_suffix.npz`
edgel3.process_file(audio_filepath, model=model, output_dir='/different/dir', suffix=
↪'suffix')
```

2.3 Using the Command Line Interface (CLI)

To compute embeddings for a single file via the command line run:

```
$ edgel3 /path/to/file.wav
```

This will create an output file at `/path/to/file.npz`.

You can change the output directory as follows:

```
$ edgel3 /path/to/file.wav --output /different/dir
```

This will create an output file at `/different/dir/file.npz`.

You can also provide multiple input files:

```
$ edgel3 /path/to/file1.wav /path/to/file2.wav /path/to/file3.wav
```

which will create the output files `/different/dir/file1.npz`, `/different/dir/file2.npz`, and `different/dir/file3.npz`.

You can also provide one (or more) directories to process:

```
$ edgel3 /path/to/audio/dir
```

This will process all supported audio files in the directory, though it will not recursively traverse the directory (i.e. audio files in subfolders will not be processed).

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

```
$ edgel3 /path/to/file.wav --suffix somesuffix
```

which will create the output file `/path/to/file_somesuffix.npz`.

To get embedding out of a *sea* model, model_type and emb_dim can be provided

```
$ edgel3 /path/to/file.wav --model-type sea --emb-dim 256
```

To get embedding out of a *sparse* model, sparsity and retrain_type arguments can be provided, for example:

```
$ edgel3 /path/to/file.wav --model-type sparse --model-sparsity 53.5 --retrain-type kd
```

By default, edgel3 will pad the beginning of the input audio signal by 0.5 seconds (half of the window size) so that the the center of the first window corresponds to the beginning of the signal, and the timestamps correspond to the center of each window. You can disable this centering as follows:

```
$ edgel3 /path/to/file.wav --no-centering
```

In the following example we change the hop size from 0.1 (10 frames per second) to 0.5 (2 frames per second):

```
$ edgel3 /path/to/file.wav --hop-size 0.5
```

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

```
$ edgel3 /path/to/file.wav --quiet
```

A sample of full command for *sparse* model may look like:

```
$ edgel3 /path/to/file.wav --output /different/dir --suffix somesuffix --model-type␣
↪sparse --model-sparsity 53.5 --retrain-type kd --no-centering --hop-size 0.5 --quiet
```

A sample of full command for *sea* model may look like:

```
$ edgel3 /path/to/file.wav --output /different/dir --suffix somesuffix --model-type␣
↪sea --emb-dim 64 --no-centering --hop-size 0.5 --quiet
```

edgel3 is an open-source Python library for downloading smaller L3 models and computing deep audio embeddings from such models for edge applications.

[1] Specialized Embedding Approximation for Edge Intelligence: A case study in Urban Sound Classification

Sangeeta Srivastava, Dhrubojyoti Roy, Mark Cartwright, Juan Pablo Bello, and Anish Arora. To be published in IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Canada, June 2021.

[2] EdgeL3: Compressing L3-Net for Mote-Scale Urban Noise Monitoring

Sangeeta Kumari, Dhrubojyoti Roy, Mark Cartwright, Juan Pablo Bello, and Anish Arora. IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Rio de Janeiro, Brazil, May 2019.

CHAPTER 3

API Reference

3.1 Modules

3.1.1 edgel3.cli

edgel3.cli.**run**(*inputs*, *output_dir=None*, *suffix=None*, *model_type='sparse'*, *emb_dim=128*, *retrain_type='ft'*, *sparsity=95.45*, *center=True*, *hop_size=0.1*, *verbose=False*)

Computes and saves L3 embedding for given inputs.

Parameters

- **inputs** (*list of str, or str*) – File/directory path or list of file/directory paths to be processed
- **output_dir** (*str or None*) – Path to directory for saving output files. If None, output files will be saved to the directory containing the input file.
- **suffix** (*str or None*) – String to be appended to the output filename, i.e. <base filename>_<suffix>.npy. If None, then no suffix will be added, i.e. <base filename>.npy.
- **model_type** (*{sea, sparse}*) – Type of smaller version of L3 model. If `sea` is selected, the audio model is a UST specialized (SEA) model. `sparse` gives a sparse L3 model with the desired `sparsity`.
- **emb_dim** (*{512, 256, 128, 64}*) – Desired embedding dimension of the UST specialized embedding approximated (SEA) models.
- **retrain_type** (*str*) – Type of retraining after sparsification of the L3 audio. Finetuned model is returned for `ft` and `kd` gives knowledge distilled sparse audio.
- **sparsity** (*{95.45, 53.5, 63.5, 72.3, 87.0}*) – The desired sparsity to be achieved for the audio model of L3. Sparsity of 95.45 corresponds to the EdgeL3 model.
- **center** (*boolean*) – If True, pads beginning of signal so timestamps correspond to center of window.
- **hop_size** (*float*) – Hop size in seconds.

- **quiet** (*boolean*) – If True, suppress all non-error output to stdout

3.1.2 edgel3.core

edgel3.core.**_center_audio**(*audio*, *frame_len*)
Center audio so that first sample will occur in the middle of the first frame

edgel3.core.**_pad_audio**(*audio*, *frame_len*, *hop_len*)
Pad audio if necessary so that all samples are processed

edgel3.core.**get_embedding**(*audio*, *sr*, *model=None*, *model_type='sparse'*, *emb_dim=128*, *retrain_type='ft'*, *sparsity=95.45*, *center=True*, *hop_size=0.1*, *verbose=1*)
Computes and returns L3 embedding for an audio data from pruned audio model.

Parameters

- **audio** (*np.ndarray [shape=(N,) or (N,C)]*) – 1D numpy array of audio data.
- **sr** (*int*) – Sampling rate, if not 48kHz or 8kHz will audio will be resampled for *sparse* and *sea* models respectively.
- **model** (*keras.models.Model or None*) – Loaded model object. If a model is provided, then *sparsity* will be ignored. If None is provided, the desired version of smaller L3 will be loaded, determined by *model_type*. model will be loaded using
- **model_type** (*{'sea', 'sparse'}*) – Type of smaller version of L3 model. If *sea* is selected, the audio model is a UST specialized (SEA) model. *sparse* gives a sparse L3 model with the desired 'sparsity'.
- **emb_dim** (*{512, 256, 128, 64}*) – Desired embedding dimension of the UST specialized embedding approximated (SEA) models. Not used for *sparse* models.
- **retrain_type** (*{'ft', 'kd'}*) – Type of retraining for the sparsified weights of L3 audio model. *ft* chooses the fine-tuning method and *kd* returns knowledge distilled model.
- **sparsity** (*{95.45, 53.5, 63.5, 72.3, 87.0}*) – The desired sparsity of audio model.
- **center** (*boolean*) – If True, pads beginning of signal so timestamps correspond to center of window.
- **hop_size** (*float*) – Hop size in seconds.
- **verbose** (*0 or 1*) – Keras verbosity.

Returns

- **embedding** (*np.ndarray [shape=(T, D)]*) – Array of embeddings for each window.
- **timestamps** (*np.ndarray [shape=(T,)]*) – Array of timestamps corresponding to each embedding in the output.

edgel3.core.**get_output_path**(*filepath*, *suffix*, *output_dir=None*)

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** – Path to output file.

Return type str

edgel3.core.**process_file**(*filepath*, *output_dir=None*, *suffix=None*, *model=None*, *model_type='sparse'*, *emb_dim=128*, *sparsity=95.45*, *center=True*, *hop_size=0.1*, *verbose=True*)

Computes and saves L3 embedding for given audio file

Parameters

- **filepath** (*str*) – Path to WAV file to be processed.
- **output_dir** (*str or None*) – Path to directory for saving output files. If None, output files will be saved to the directory containing the input file.
- **suffix** (*str or None*) – String to be appended to the output filename, i.e. <base filename>_<suffix>.npz. If None, then no suffix will be added, i.e. <base filename>.npz.
- **model** (*keras.models.Model or None*) – Loaded model object. If a model is provided, then *model_type* will be ignored. If None is provided, UST specialized L3 or sparse L3 is loaded according to the `model_type`.
- **model_type** (*{'sea', 'sparse'}*) – Type of smaller version of L3 model. If *sea* is selected, the audio model is a UST specialized (SEA) model. *sparse* gives a sparse L3 model with the desired 'sparsity'.
- **emb_dim** (*{512, 256, 128, 64}*) – Desired embedding dimension of the UST specialized embedding approximated (SEA) models. Not used for *sparse* models.
- **sparsity** (*{95.45, 53.5, 63.5, 72.3, 87.0}*) – The desired sparsity of audio model.
- **center** (*boolean*) – If True, pads beginning of signal so timestamps correspond to center of window.
- **hop_size** (*float*) – Hop size in seconds.
- **verbose** (*0 or 1*) – Keras verbosity.

3.1.3 edgel3.models

edgel3.models.**_construct_sparsified_audio_network**(***kwargs*)

Returns an uninitialized model object for a sparsified network with a Melspectrogram input (with 256 frequency bins).

Returns **model** – Model object.

Return type keras.models.Model

edgel3.models.**_construct_ust_specialized_audio_network**(*emb_dim=128*, ***kwargs*)

Returns an uninitialized model object for a UST specialized audio network with a Melspectrogram input (with 64 frequency bins).

Returns **model** – Model object.

Return type keras.models.Model

edgel3.models.**load_embedding_model**(*model_type*, *emb_dim*, *retrain_type*, *sparsity*)

Returns a model with the given characteristics. Loads the model if the model has not been loaded yet.

Parameters

- **model_type** (*{sea, sparse}*) – Type of smaller version of L3 model. If 'sea' is selected, the audio model is a UST specialized (SEA) model. 'sparse' gives a sparse L3 model with the desired 'sparsity'.
- **emb_dim** (*{512, 256, 128, 64}*) – Desired embedding dimension of the UST specialized embedding approximated (SEA) models.
- **retrain_type** (*'ft' or 'kd'*) – Type of retraining for the sparsified weights of L3 audio model. 'ft' chooses the fine-tuning method and 'kd' returns knowledge distilled model.
- **sparsity** (*{95.45, 53.5, 63.5, 72.3, 87.0}*) – The desired sparsity of audio model.

Returns **model** – Model object.

Return type keras.models.Model

edgel3.models.**load_embedding_model_path**(*model_type*, *emb_dim*, *retrain_type*, *sparsity*)

Returns the local path to the model weights file for the model with the given sparsity

Parameters

- **model_type** (*{sea, sparse}*) – Type of smaller version of L3 model. If 'sea' is selected, the audio model is a UST specialized (SEA) model. 'sparse' gives a sparse L3 model with the desired 'sparsity'.
- **emb_dim** (*{512, 256, 128, 64}*) – Desired embedding dimension of the UST specialized embedding approximated (SEA) models.
- **retrain_type** (*'ft' or 'kd'*) – Type of retraining for the sparsified weights of L3 audio model. 'ft' chooses the fine-tuning method and 'kd' returns knowledge distilled model.
- **sparsity** (*{95.45, 53.5, 63.5, 72.3, 87.0}*) – Desired sparsity of the audio model.

Returns **output_path** – Path to given model object

Return type str

CHAPTER 4

Citations for edgel3

[1] **Specialized Embedding Approximation for Edge Intelligence: A case study in Urban Sound Classification** Sangeeta Srivastava, Dhrubojyoti Roy, Mark Cartwright, Juan Pablo Bello, and Anish Arora. To be published in IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Canada, June 2021.

[2] **EdgeL3: Compressing L3-Net for Mote-Scale Urban Noise Monitoring** Sangeeta Kumari, Dhrubojyoti Roy, Mark Cartwright, Juan Pablo Bello, and Anish Arora. IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Rio de Janeiro, Brazil, May 2019.

[3] **Look, Listen and Learn More: Design Choices for Deep Audio Embeddings** Jason Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), pages 3852-3856, Brighton, UK, May 2019.

[4] **Look, Listen and Learn** Relja Arandjelović and Andrew Zisserman IEEE International Conference on Computer Vision (ICCV), Venice, Italy, Oct. 2017.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index

Symbols

E

G

L

P

R