GluonAR Documentation

Release 0.1.0

haoxintong

Jun 24, 2019

Contents

Gluon Audio is a toolkit providing deep learning based audio recognition algorithm. The project is still under development, and only Chinese introduction will be provided.

CHAPTER 1

GluonAR Introduction:

GluonAR is based on MXnet-Gluon, if you are new to it, please check out dmlc 60-minute crash course.

GluonAR, Text-Independent Speaker Recognition.

feature:

- ffmpegpythonic binding `avlibrosa`audio
- **`Hybridize()`. forwardpysound, librosa, scipy, , end-to-end, :**
  - `nd.contrib.fft(STFTBlock)`z-score block, numpyscipyGPU12%.
  - `MelSpectrogram`, `DCT1D`, `MFCC`, `PowerToDB`
  - 1808.00158`SincBlock`
- gluonVOX
- Speaker Verification
- , VOX11:1acc: 0.941152+-0.004926

example:

```
import numpy as np
import mxnet as mx
import librosa as rosa
from gluonar.utils.viz import view_spec
from gluonar.nn.basic_blocks import STFTBlock

data = rosa.load(r"resources/speaker_recognition/speaker0_0.m4a", sr=16000)[0][:35840]
nd_data = mx.nd.array([data], ctx=mx.gpu())

stft = STFTBlock(35840, hop_length=160, win_length=400)
stft.initialize(ctx=mx.gpu())

# stft block forward
ret = stft(nd_data).asnumpy()[0][0]
```

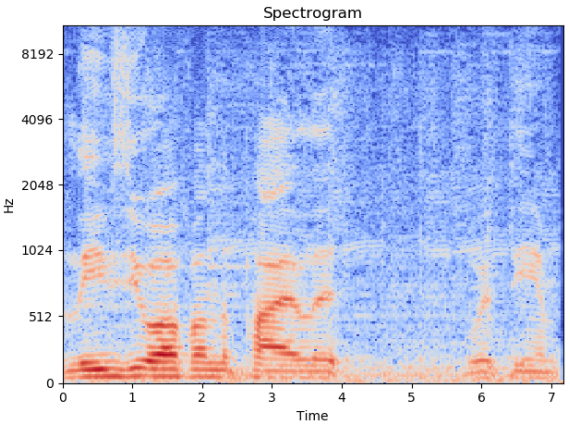
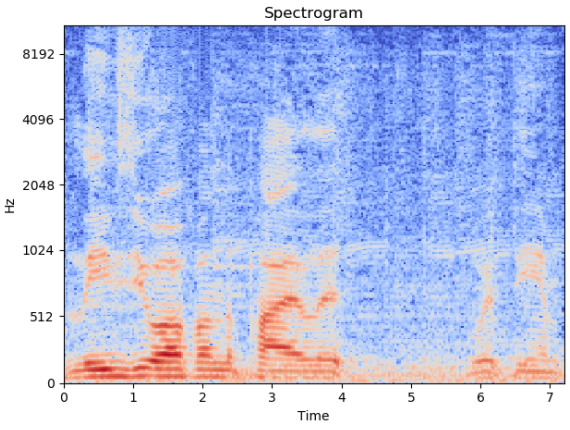
(continues on next page)

(continued from previous page)

```
spec = np.transpose(ret, (1, 0)) ** 2
view_spec(spec)

# stft in librosa
spec = rosa.stft(data, hop_length=160, win_length=400, window="hamming")
spec = np.abs(spec) ** 2
view_spec(spec)
```

:

STFTBlock	STFT in librosa
	

examples/.

1.1 Install

1.1.1 Requirements

mxnet-1.5.0+, gluonfr, av, librosa, . . .

, , librosaaac pyav8.

- librosa

```
pip install librosa
```

- ffmpeg

```
# ffmpeg,
./configure --extra-cflags=-fPIC --enable-shared
make -j
sudo make install
```

- pyav, ffmpeg

```
pip install av
```

- gluonfr

```
pip install git+https://github.com/THUFutureLab/gluon-face.git@master
```

1.2 Datasets

1.2.1 TIMIT

The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT) Training and Test Data. Before using this dataset please follow the instruction on link.

A copy of this was uploaded to Google Drive by @philipperemy here.

1.2.2 VoxCeleb

VoxCeleb is an audio-visual dataset consisting of short clips of human speech, extracted from interview videos uploaded to YouTube.

For more information, checkout this page.

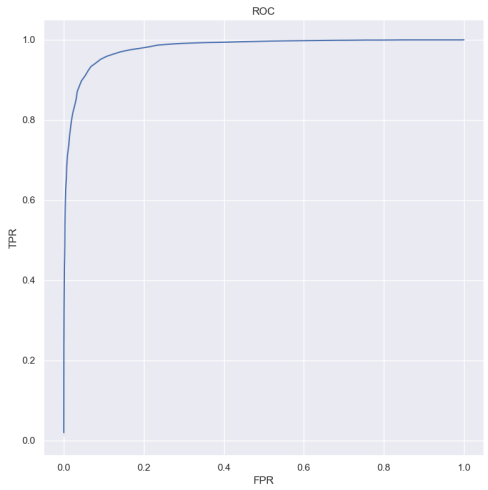
1.3 Pretrained Models

1.3.1 Speaker Recognition

ResNet18 training with VoxCeleb

Download: Baidu, Google Drive

I followed the ideas in paper **VoxCeleb2** 1806.05622 to train this model, the differences between them:

	Res18 in this repo	Res34 in paper
Trained on	VoxCeleb2	VoxCeleb2
Input spec size	224x224	512x300
Eval on	Random 9500+ pair samples from VoxCeleb1	Original VoxCeleb1 test set
Metric	Accuracy: 0.932656 +- 0.005187	EER: 0.0504
Framework	Mxnet Gluon	Matconvnet
ROC		

1.4 GluonAR API

1.4.1 gluonar.data

This module provides popular audio recognition datasets.

Hint: Please refer to *Datasets* for the description of the datasets listed in this page, and how to download and extract them.

API Reference

class gluonar.data.**VoxAudioFolderDataset**

Load an audio file .

Parameters

- **root** (*string*) – path to vox root.
- **sr** (*int, default is 16k.*) – target sampling rate
- **min_length** (*int, default is 3.*) – min length audio required.

1.4.2 gluonar.data.transforms

This file includes various transformations that are critical to audio tasks.

API Reference

Some functions about audio processing and transform.

1.4.3 gluonar.model_zoo

Models for audio recognition

gluonar.model_zoo.get_model

Returns a pre-defined GluonAR model by name.

Hint: This is the recommended method for getting a pre-defined model.

```
get_model
```

API Reference

Models for audio recognition

1.4.4 gluonar.nn

Neural Network Components.

Hint: Not every component listed here is HybridBlock, which means some of them are not hybridizable. However, we are trying our best to make sure components required during inference are hybridizable so the entire network can be exported and run in other languages.

For example, encoders are usually non-hybridizable but are only required during training. In contrast, decoders are mostly 'HybridBlock's.

Basic Blocks

Blocks that usually used in audio processing.

SincConv1D	Sinc Conv Block from "Speaker Recognition from Raw Waveform with SincNet" paper.
ZScoreNormBlock	Zero Score Normalization Block
STFTBlock	Short-Time Fourier Transform Block.
DCT1D	Compute the Discrete Cosine Transform of input data.
MelSpectrogram	Compute a mel-scaled spectrogram.
MFCC	Mel-frequency cepstral coefficients (MFCCs)
PowerToDB	Convert a power spectrogram (amplitude squared) to decibel (dB) units.

API Reference

Basic Blocks used in GluonAR.

class gluonar.nn.basic_blocks.**SincConv1D**
Sinc Conv Block from "Speaker Recognition from Raw Waveform with SincNet" paper.

class gluonar.nn.basic_blocks.**ZScoreNormBlock**
Zero Score Normalization Block

class gluonar.nn.basic_blocks.**STFTBlock**
Short-Time Fourier Transform Block.

Parameters

- **audio_length** (*int.*) – target audio length.
- **n_fft** (*int > 0 [scalar]*) – length of the FFT window
- **hop_length** (*int > 0 [scalar]*) – number of samples between successive frames. See *librosa.core.stft*
- **win_length** (*int <= n_fft [scalar]*) – Each frame of audio is windowed by *window()*. The window will be of length *win_length* and then padded with zeros to match *n_fft*. If unspecified, defaults to `win_length = n_fft`.
- **window** (*string [shape=(n_fft,)]*) – A window specification (string, tuple, or number), see *scipy.signal.get_window*
- **center** (*boolean*) – If *True*, the signal *y* is padded so that frame *D[:, t]* is centered at *y[t * hop_length]*. If *False*, then *D[:, t]* begins at *y[t * hop_length]*
- **power** (*float > 0 [scalar]*) – Exponent for the magnitude melspectrogram. e.g., 1 for energy, 2 for power, etc.

- **Inputs:**
  - **x**: the input audio signal, with shape `(batch_size, audio_length)`.
- **Outputs:**
  - **specs**: specs tensor with shape `(batch_size, 1, num_frames, n_fft/2)`.

Notes

The num_frames is calculated by `1+(len(y)-n_fft)/hop_length` when center is True, and different from librosa the output should be transposed before visualization.

class gluonar.nn.basic_blocks.**DCT1D**
Compute the Discrete Cosine Transform of input data. This block is implemented as scipy.

DCT1D's behavior is compute dct along last axis for any dimensions larger than 2.

Parameters

- **mode** (*{1, 2, 3}, optional.*) – Type of the DCT (see Notes). Default type is 2.
- **N** (*int.*) – Length of the transform. The required value is `N = x.shape[axis]`.
- **norm** (*{None, 'ortho'}, optional.*) – Normalization mode (see Notes). Default is None.

Notes

Type II

There are several definitions of the DCT-II; use scipy definition following (for `norm=None`):

```
           N-1
y[k] = 2* sum x[n]*cos(pi*k*(2n+1)/(2*N)), 0 <= k < N.
           n=0
```

If `norm='ortho'`, `y[k]` is multiplied by a scaling factor *f*:

```
f = sqrt(1/(4*N)) if k = 0,
f = sqrt(1/(2*N)) otherwise.
```

Which makes the corresponding matrix of coefficients orthonormal (`OO' = Id`).

class gluonar.nn.basic_blocks.**MelSpectrogram**

Compute a mel-scaled spectrogram.

Parameters

- **audio_length** (*int.*) – target audio length.
- **sr** (*number > 0 [scalar]*) – sampling rate of *audio*
- **n_fft** (*int > 0 [scalar]*) – length of the FFT window
- **hop_length** (*int > 0 [scalar]*) – number of samples between successive frames. See *librosa.core.stft*
- **power** (*float > 0 [scalar]*) – Exponent for the magnitude melspectrogram. e.g., 1 for energy, 2 for power, etc.
- **others** (*additional arguments*) – Mel filter bank parameters. See *librosa.filters.mel* for details.

class gluonar.nn.basic_blocks.**MFCC**

Mel-frequency cepstral coefficients (MFCCs)

Parameters

- **audio_length** (*int.*) – target audio length.
- **sr** (*number > 0 [scalar]*) – sampling rate of *y*
- **n_mfcc** (*int > 0 [scalar]*) – number of MFCCs to return
- **dct_type** (*None, or {1, 2, 3}*) – Discrete cosine transform (DCT) type. Now only DCT type-2 is used.
- **norm** (*None or 'ortho'*) – If *dct_type* is *2 or 3*, setting *norm='ortho'* uses an ortho-normal DCT basis. Normalization is not supported for *dct_type=1*.

See also:

`librosa.melspectrogram`, `scipy.fftpack.dct`

class gluonar.nn.basic_blocks.**PowerToDB**

Convert a power spectrogram (amplitude squared) to decibel (dB) units. This is modified from librosa.power_to_db, and make it be able to process batch input.

For input shape of (batch, channel, w, h) or (batch, w, h), this block will compute power to db along last 2 axis.

Parameters

- **ref** (*float.*) – The amplitude *abs(S)* is scaled relative to *ref*: *10 * log10(S / ref)*.
- **amin** (*float > 0 [scalar]*) – minimum threshold for *abs(S)* and *ref*
- **top_db** (*float >= 0 [scalar]*) – threshold the output at *top_db* below the peak: `max(10 * log10(S)) - top_db`

1.4.5 gluonar.loss

Custom losses. Losses are subclasses of gluon.loss.SoftmaxCrossEntropyLoss which is a HybridBlock actually.

gluonar.loss.ArcLoss	ArcLoss from "ArcFace: Additive Angular Margin Loss for Deep Face Recognition" paper.
gluonar.loss.RingLoss	Computes the Ring Loss from "Ring loss: Convex Feature Normalization for Face Recognition" paper.

API Reference

Custom losses. Losses are subclasses of gluon.loss.SoftmaxCrossEntropyLoss which is a HybridBlock actually.

gluonar.loss.**get_loss**(*name*, ***kwargs*)

Parameters

- **name** (*str*) – Loss name, check gluonar.loss for details.
- **kwargs** (*str*) – Params

Returns The loss.

Return type HybridBlock

class gluonar.loss.**ArcLoss**

ArcLoss from "ArcFace: Additive Angular Margin Loss for Deep Face Recognition" paper.

Parameters

- **classes** (*int.*) – Number of classes.
- **m** (*float.*) – Margin parameter for loss.
- **s** (*int.*) – Scale parameter for loss.

- **Outputs:**
  - **loss**: loss tensor with shape (batch_size,). Dimensions other than batch_axis are averaged out.

class gluonar.loss.**RingLoss**

Computes the Ring Loss from "Ring loss: Convex Feature Normalization for Face Recognition" paper.

$$L = -\sum_{i} \log(pred)_{i,label_i} + \frac{\lambda}{2m} \sum_{i=1}^{m} (\|\mathcal{F}(x_i)\|_2 - R)^2$$

Parameters **lamda** (*float*) – The loss weight enforcing a trade-off between the softmax loss and ring loss.

- **Outputs:**
  - **loss**: loss tensor with shape (batch_size,). Dimensions other than batch_axis are averaged out.

1.4.6 gluonar.utils

We implemented a broad range of utility functions which cover visualization, file handler, download and training helpers.

Visualization

plot_accuracy
plot_roc
view_spec

API Reference

Visualization tools for gluonar

Python Module Index

g

Index