torchiva

Release 0.0.1

Robin Scheibler, Kohei Saijo

Jan 30, 2023

CONTENTS:

A package for blind source separation and beamforming in pytorch .

- supports many BSS and beamforming methods
- supports memory efficient gradient computation for training neural source models
- supports batched computations
- can run on GPU via pytorch

CHAPTER

ONE

QUICK START

The package can be installed via pip:

```
pip install torchiva
```

CHAPTER

TWO

SEPARATION USING PRE-TRAINED MODEL

We provide a pre-trained model in *trained_models/tiss*. You can easily try separation with the pre-trained model:

```
# Separation
python -m torchiva.separate INPUT OUTPUT
```

where `INPUT` is either a multichannel wav file or a folder containing multichannel wav files. If a folder, then all the files inside are separted. The output is saved to `OUTPUT`. The model stored in `trained_models/tiss` is automatically downloaded to `$HOME/.torchiva_models`. The path or url to the model can also be manually provided via the `--model` option. The model was trained on the WSJ1-mix dataset with the same configuration as `./examples/configs/tiss.json`.

CHAPTER

THREE

TRAINING

We provide some simple training scripts. We support training of **T-ISS**, **MWF**, **MVDR**, **GEV**:

```
cd examples

# install some modules necessary for training
pip install -r requirements.txt

# training
python train.py PATH_TO_CONFIG PATH_TO_DATASET
```

Note that our example scripts assumes using WSJ1-mix dataset. If you want to use other datasets, please change the script in the part that loads audios.

Test your trained model with checkpoint from epoch 128:

```
# python ./test.py --dataset ../wsj1_6ch --n_fft 2048 --hop 512 --n_iter 40 --iss-
↪hparams checkpoints/tiss_delay1tap5_2ch/lightning_logs/version_0/hparams.yaml --epoch␣
↪128 --test
```

Export the trained model for later use:

```
python ./export_model.py ../trained_models/tiss checkpoints/tiss_delay1tap5_2ch/
↪lightning_logs/version_0 128 146 148 138 122 116 112 108 104 97
```

Run the example script using the exported model:

```
python ./example_dnn.py ../wsj1_6ch ../trained_models/tiss -m 2 -r 100
```

CHAPTER

FOUR

AUTHORS

- Robin Scheibler
- Kohei Saijo

CHAPTER

FIVE

LICENSE

CHAPTER

SIX

API

6.1 T_ISS

class torchiva.**T_ISS**(*n_iter=10*, *n_taps=0*, *n_delay=0*, *n_src=None*, *model=None*, *proj_back_mic=0*, *use_dmc=False*, *eps=None*)

Joint dereverberation and separation with *time-decorrelation iterative source steering* (T-ISS)[1].

Parameters can also be specified during a forward call. In this case, the forward argument is only used in that forward process and **does not rewrite class attributes**.

Parameters

- **n_iter** (*int, optional*) – The number of iterations. (default: `10`)
- **n_taps** (*int, optional*) – The length of the dereverberation filter. If set to `0`, this method works as the normal AuxIVA with ISS update[2] (default: `0`).
- **n_delay** (*int, optional*) – The number of delay for dereverberation (default: `0`).
- **n_src** (*int, optional*) – The number of sources to be separated. When `n_src` < `n_chan`, a computationally cheaper variant (Over-T-ISS)[3] is used. If set to `None`, `n_src` is set to `n_chan` (default: `None`)
- **model** (*torch.nn.Module, optional*) – The model of source distribution. Mask estimation neural network can also be used. If `None`, spherical Laplace is used (default: `None`).
- **proj_back_mic** (*int, optional*) – The reference mic index to perform projection back. If set to `None`, projection back is not applied (default: `0`).
- **use_dmc** (*bool, optonal*) – If set to `True`, memory efficient Demixing Matrix Checkpointing (DMC)[4] is used to compute the gradient. It reduces the memory cost to that of a single iteration when training neural source model (default: `False`).
- **eps** (*float, optional*) – A small constant to make divisions and the like numerically stable (default:`None`).

forward(*n_iter=None*, *n_taps=None*, *n_delay=None*, *n_src=None*, *model=None*, *proj_back_mic=None*, *use_dmc=None*, *eps=None*)

[1] T. Nakashima, R. Scheibler, M. Togami, and N. Ono, "Joint dereverberation and separation with iterative source steering", ICASSP, 2021, https://arxiv.org/pdf/2102.06322.pdf.

[2] R. Scheibler, and N Ono, "Fast and stable blind source separation with rank-1 updates" ICASSP, 2021,

[3] R. Scheibler, W. Zhang, X. Chang, S. Watanabe, and Y. Qian, "End-to-End Multi-speaker ASR with Independent Vector Analysis", arXiv preprint arXiv:2204.00218, 2022, https://arxiv.org/pdf/2204.00218.pdf.

[4] K. Saijo, and R. Scheibler, "Independence-based Joint Speech Dereverberation and Separation with Neural Source Model", arXiv preprint arXiv:2110.06545, 2022, https://arxiv.org/pdf/2110.06545.pdf.

Parameters
X (*torch.Tensor*) – The input mixture in STFT-domain, `shape (..., n_chan, n_freq, n_frames)`

Returns
Y – The separated and dereverberated signal in STFT-domain

Return type
torch.Tensor, `shape (..., n_src, n_freq, n_frames)`

Note:

This class can handle various BSS methods with ISS update rule depending on the specified arguments:

- IVA-ISS: `n_taps=0, n_delay=0, n_chan==n_src, model=LaplaceMoldel()` or `GaussMoldel()`
- ILRMA-ISS: `n_taps=0, n_delay=0, n_chan==n_src, model=NMFModel()`
- DNN-IVA-ISS: `n_taps=0, n_delay=0, n_chan==n_src, model=*DNN*`
- OverIVA-ISS: `n_taps=0, n_delay=0, n_chan < n_src`
- ILRMA-T-ISS[Page 13, 1] : `n_taps>0, n_delay>0, n_chan==n_src, model=NMFMoldel()`
- DNN-T-ISS[Page 13, 4] : `n_taps>0, n_delay>0, n_chan==n_src, model=*DNN*`
- Over-T-ISS[Page 13, 3] : `n_taps>0, n_delay>0, n_chan > n_src`

References

6.2 AuxIVA_IP

class torchiva.**AuxIVA_IP**(*n_iter=10*, *n_src=None*, *model=None*, *proj_back_mic=0*, *eps=None*)

Independent vector analysis (IVA) with iterative projection (IP) update[5].

We do not support ILRMA-T with IP updates.

Parameters

- **n_iter** (*int, optional*) – The number of iterations. (default: `10`)
- **n_src** (*int, optional*) – The number of sources to be separated. When `n_src < n_chan`, a computationally cheaper variant (OverIVA)[6] is used. If set to `None`, `n_src` is set to `n_chan` (default: `None`)
- **model** (*torch.nn.Module, optional*) – The model of source distribution. If `None`, spherical Laplace is used (default: `None`).
- **proj_back_mic** (*int, optional*) – The reference mic index to perform projection back. If set to `None`, projection back is not applied (default: `0`).
- **eps** (*float, optional*) – A small constant to make divisions and the like numerically stable (default:`None`).

[5] N. Ono, "Stable and fast update rules for independent vector analysis based on auxiliary function technique", WASSPA, 2011.
[6] R. Scheibler, and N Ono, "Independent vector analysis with more microphones than sources", WASSPA, 2019, https://arxiv.org/pdf/1905.07880.pdf.

forward(*X*, *n_iter=None*, *n_src=None*, *model=None*, *proj_back_mic=None*, *eps=None*)

Parameters
X (*torch.Tensor*) – The input mixture in STFT-domain, `shape (..., n_chan, n_freq, n_frames)`

Returns
Y – The separated signal in STFT-domain

Return type
torch.Tensor, `shape (..., n_src, n_freq, n_frames)`

Note:

This class can handle two BSS methods with IP update rule depending on the specified arguments:

- AuxIVA-IP: `n_chan==n_src, model=LaplaceMoldel() or GaussMoldel()`
- ILRMA-IP: `n_chan==n_src, model=NMFModel()`
- OverIVA_IP[6]: `n_taps=0, n_delay=0, n_chan==n_src, model=NMFModel()`

References

6.3 AuxIVA_IP2

class torchiva.**AuxIVA_IP2**(*n_iter=10*, *model=None*, *proj_back_mic=0*, *eps=None*)

Blind source separation based on independent vector analysis with alternating updates of the mixing vectors[7]

Parameters

- **n_iter** (*int, optional*) – The number of iterations (default: `10`).
- **model** (*torch.nn.Module, optional*) – The model of source distribution. If `None`, spherical Laplace is used (default: `None`).
- **proj_back_mic** (*int, optional*) – The reference mic index to perform projection back. If set to `None`, projection back is not applied (default: `0`).
- **eps** (*float, optional*) – A small constant to make divisions and the like numerically stable (default:`None`).

forward(*X*, *n_iter=None*, *model=None*, *proj_back_mic=None*, *eps=None*)

Parameters
X (*torch.Tensor*) – The input mixture in STFT-domain, `shape (..., n_chan, n_freq, n_frames)`

Returns
X – The separated signal in STFT-domain.

Return type
torch.Tensor, `shape (..., n_chan, n_freq, n_frames)`

[7] N. Ono, "Fast stereo independent vector analysis and its implementation on mobile phone", IWAENC, 2012.

References

6.4 WPE

class torchiva.**WPE**(*n_iter=3*, *n_delay=3*, *n_taps=5*, *model=None*, *eps=1e-05*)

Weighted prediction error (WPE)[9].

Parameters

- **n_iter** (*int, optional*) – The number of iterations. (default: 3)
- **n_taps** (*int, optional*) – The length of the dereverberation filter (default: 5).
- **n_delay** (*int, optional*) – The number of delay for dereverberation (default: 3).
- **model** (*torch.nn.Module, optional*) – The model of source distribution. If None, time-varying Gaussian is used. (default: None).
- **eps** (*float, optional*) – A small constant to make divisions and the like numerically stable (default:1e-5).

Returns

Y – The dereverberated signal in STFT-domain.

Return type

torch.Tensor, shape (..., n_src, n_freq, n_frames)

References

6.5 MVDR beamformer

class torchiva.**MVDRBeamformer**(*mask_model*, *ref_mic=0*, *eps=1e-05*, *mvdr_type='rtf'*, *n_power_iter=None*)

Implementation of MVDR beamformer. This class is basically assumes DNN-based beamforming. also supports the case of estimating three masks

Parameters

- **mask_model** (*torch.nn.Module*) – A function that is given one spectrogram and returns 2 or 3 masks of the same size as the input. When 3 masks (1 for target and the rest 2 for noise) are etimated, they are utilized as in[10]
- **ref_mic** (*int, optional*) – Reference channel (default: 0)
- **eps** (*float, optional*) – A small constant to make divisions and the like numerically stable (default:1e-5).
- **mvdr_type** (*str, optional*) – The way to obtain the MVDR weight. If set to rtf, relative transfer function is computed to obtain MVDR. If set to 'scm', MVDR weight is obtained directly with spatial covariance matrices[11] (default: rtf).

[9] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B. H. Juang, "Speech dereverberation based on variance-normalized delayed linear prediction", IEEE Trans. on Audio, Speech, and Lang. Process., 2010.

[10] C. Boeddeker et al., "Convolutive Transfer Function Invariant SDR training criteria for Multi-Channel Reverberant Speech Separation", ICASSP, 2021.

[11] Mehrez Souden, Jacob Benesty, and Sofiene Affes, "On optimal frequency-domain multichannel linear filtering for noise reduction", IEEE Trans. on audio, speech, and lang. process., 2009.

- **n_power_iter** (*int, optional*) – Use the power iteration method to compute the relative transfer function instead of the full generalized eigenvalue decomposition (GEVD). The number of iteration desired should be provided. If set to None, the full GEVD is used (default: None).

forward(*X*, *mask_model=None*, *ref_mic=None*, *eps=None*, *mvdr_type=None*, *n_power_iter=None*)

Parameters
X (*torch.Tensor*) – The input mixture in STFT-domain, shape (..., n_chan, n_freq, n_frames)

Returns
Y – The separated signal in STFT-domain

Return type
torch.Tensor, shape (..., n_src, n_freq, n_frames)

References

6.6 MWF beamformer

class torchiva.**MWFBeamformer**(*mask_model*, *ref_mic=0*, *eps=1e-05*, *time_invariant=True*)

Implementation of MWF beamformer described in[12]. This class is basically assumes DNN-based beamforming.

Parameters

- **mask_model** (*torch.nn.Module*) – A function that is given one spectrogram and returns 2 masks of the same size as the input.
- **ref_mic** (*int, optional*) – Reference channel (default: 0)
- **eps** (*float, optional*) – A small constant to make divisions and the like numerically stable (default:1e-5).
- **time_invariant** (*bool, optional*) – If set to True, this flag indicates that we want to use the time-invariant version of MWF. If set to False, the time-varying MWF is used instead (default: True).

forward(*X*, *mask_model=None*, *ref_mic=None*, *eps=None*, *time_invariant=None*)

Parameters
X (*torch.Tensor*) – The input mixture in STFT-domain, shape (..., n_chan, n_freq, n_frames)

Returns
Y – The separated signal in STFT-domain

Return type
torch.Tensor, shape (..., n_src, n_freq, n_frames)

[12] Y. Masuyama et al., "Consistency-aware multi-channel speech enhancement using deep neural networks", ICASSP, 2020.

References

6.7 GEV beamformer

class torchiva.**GEVBeamformer**(*mask_model*, *ref_mic=0*, *eps=1e-05*)

Implementation of GEV beamformer. This class is basically assumes DNN-based beamforming.

Parameters

- **mask_model** (*torch.nn.Module*) – A function that is given one spectrogram and returns 2 masks of the same size as the input.
- **ref_mic** (*int, optional*) – Reference channel (default: `0`)
- **eps** (*float, optional*) – A small constant to make divisions and the like numerically stable (default:`1e-5`).

forward(*X*, *mask_model=None*, *ref_mic=None*, *eps=None*)

Parameters

X (*torch.Tensor*) – The input mixture in STFT-domain, `shape (..., n_chan, n_freq, n_frames)`

Returns

Y – The separated signal in STFT-domain

Return type

torch.Tensor, `shape (..., n_src, n_freq, n_frames)`

6.8 FIVE

class torchiva.**FIVE**(*n_iter=10*, *model=None*, *proj_back_mic=0*, *eps=None*, *n_power_iter=None*)

Fast independent vector extraction (FIVE)[8]. FIVE extracts one source from the input signal.

Parameters

- **n_iter** (*int, optional*) – The number of iterations (default: `10`).
- **model** (*torch.nn.Module, optional*) – The model of source distribution (default: `LaplaceModel`).
- **proj_back_mic** (*int, optional*) – The reference mic index to perform projection back. If set to `None`, projection back is not applied (default: `0`).
- **eps** (*float, optional*) – A small constant to make divisions and the like numerically stable (default: `None`).
- **n_power_iter** (*int, optional*) – The number of power iterations. If set to `None`, eigenvector decomposition is used instead. (default: `None`)

forward(*X*, *n_iter=None*, *model=None*, *proj_back_mic=None*, *eps=None*)

Parameters

X (*torch.Tensor*) – The input mixture in STFT-domain, `shape (..., n_chan, n_freq, n_frames)`

[8] R. Scheibler, and N Ono, "Fast independent vector extraction by iterative SINR maximization", ICASSP, 2020, https://arxiv.org/pdf/1910.10654.pdf.

Returns
Y – The extracted *one* signal in STFT-domain.

Return type
torch.Tensor, `shape (..., n_freq, n_frames)`

References

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX