3 Speaker Separation Dataset Re-Construction Training and Testing For Conv-TasNet

Author
Adam A. Whitaker-Wilson

Supervisor
Kevin O'Neil

Course
Computing Science Project
COMP 4910

Thompson Rivers University
Kamloops B.C. Canada

April 11, 2020

Contents

1 Abstract

This paper presents a dataset construction approach for 3 speaker separation of The ”Cocktail Party Problem” for Conv-TasNet neural net model implementations.

By comparing a traditional digital dataset composition with a proposed analog dataset composition approach, we were hoping to see a significant difference between them.

What is shown in this paper is a comparative analysis between 4 Conv-TasNet models, each trained on different datasets. Two of them were generated using digital method, the other two using a proposed analog mixing method. The results indicate that the proposed analog recording method, with further investigation, may be able to improve training the Conv-TasNet architecture.

2 Introduction

In recent years, the use of transcription systems have been increasing and the expected accuracy of such systems are still unmet. Applications such as Reason8, Otter.AI and Voicera do well with providing end-to-end solutions, but they all have troubles when people speak over each other.

There are 3 main phases to an end-to-end transcription application, speaker diarization, speaker separation and transcription. Although there are many successful speaker diarization and transcription solutions, speaker separation is still a ongoing bottleneck concerning overall system accuracy. When given an audio recording, the issue with ineffective speaker separation is that the diarization phase concludes that a new person is speaking and the transcription phase cannot determine what is said or makes a mistake.

An accurate automatic offline end-to-end transcription system could save a lot of time and could potentially detect nuances of a conversation or meeting such as detecting what a person was trying to say while being interrupted or hidden conversations between people in a crowded environment.

Solving the speaker separation problem could be also be useful audio forensic, restoration and analysis applications as well.

The creation of datasets in a way that can train a neural network model well for separating overlapping utterances of different speakers may contribute to increasing accuracy of existing and future models. The dataset creation methodology proposed in this paper is structured to set a common ”base” for mixing digital audio sources from multiple datasets together by projecting the audio sources through loudspeakers in a room and recording the mixture with a measurement microphone. The analog speaker projection may address the issue of the neural net model learning to separate audio sources based on digital audio source attributes such as sample rate, bit rate and bit depth rather than the spectral properties of the persons speaking.

Hopefully this paper will insight others to create similar approaches to dataset creation that will contribute to solving other audio related neural net problems.

3 Conv-TasNet Overview

As described in [9], ”An encoder maps a segment of the mixture waveform to a high-dimensional representation and a separation module calculates a multiplicative function (i.e., a mask) for each of the target sources. A decoder reconstructs the source waveforms from the masked features.”

The encoder receives the input mixture into a 1-D convolutional autoencoder that models the waveform. Based on the autoencoder output, a temporal convolution network (separation module) estimates the masks. Each block has a depthwise convolution operation and the decoder is a 1-D convolution layer which outputs the separated sources.

4 Dataset Creation Comparison - Approach & Design

For this project, two approaches to dataset creation are compared. For each approach two 3 speaker datasets were generated and trained on the Conv-TasNet speaker separation implementation offered by [8].

For each composition a folded is created and it contains 4 files, mixture, spk1, spk2, spk3, where the spk files are ground truths.

A total of 5 datasets are generated, the additional dataset is created for testing.

The first approach is composed using all digital means, where the second approach is composed by projecting the audio into a room and recording the mixtures and ground truths with a single measurement microphone.

Programming Language: Python

Framework: pytorch

Model framework: Conv-TasNet implementation by [8]

Digital Composition Function Derived by [10]

The Digital Dataset Construction Approach:

Mixtures were composed using a modified version of the dataset composition function offered in the WASS [10] audio source separation Conv-TasNet implementation.

This composition system contains a folder for audio sources to be separated (in this case human speakers) and a folder for noise (in this case ambiance, objecs, wind, etc) injection. A Gaussian noise function is also included. Compositions are generated by randomly selecting source files and ambient sources to be mixed together. Gaussian noise is also injected at a random volume. All randomization ranges are based on configuration parameters.

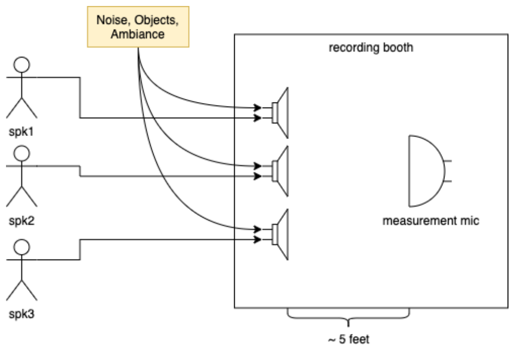
Compositions were generated based on the following configuration parameters:

Duration: 4 (seconds) Sample Rate: 8000 Label Scale: 0.5 - 1.5 Ambient Scale: 0.4 - 0.6 Signal to Noise Ratio: 0 - 100

No preprocessing was applied.

The Analog Dataset Construction Approach:

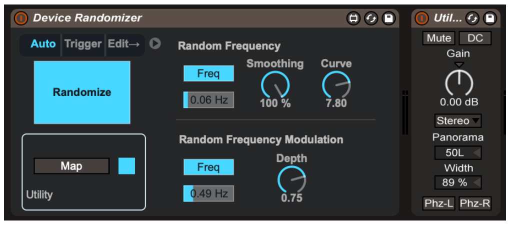
Mixtures were composed by assigning a speaker to a dedicated loudspeaker which is then projected into a room. Every loudspeaker has the same random ambience uniformly injected at random volumes. A single microphone records the mixture, followed by a clean projection of each speaker without ambience injected to be used as ground truths.



This approach was executed by using the following:

Digital Audio Workstation (DAW) Ableton Live tascam us-16x08 audio interface 3x ev-zx1i-90 loudspeakers (3 speaker separation) 1x apex220 microphone

Device Randomizer used for noise injection.



Random Frequency
Freq 0.06 Hz
Smoothing 100 %
Curve 7.80
Random Frequency Modulation
Freq 0.49 Hz
Depth 0.75
Gain 0.00 dB
Stereo
Panorama 50L
Width 89 %

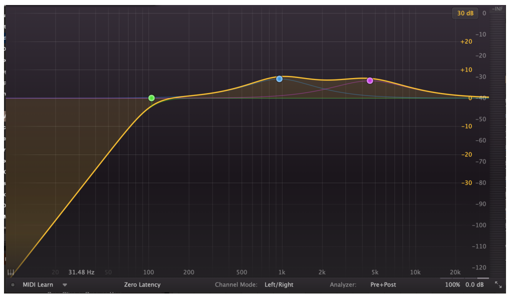
Raw Composition Attributes:

Duration: 4 (seconds) Sample Rate: 48000 Bit Rate: 2304 kbps Bit Depth: 24

4.0.1 Hardware Calibration

Each loudspeaker was calibrated by injecting pink noise and applying the necessary equalization, so the audio is projected as close to its true representation as possible.

For this particular case a 18db/octave high pass filter at 105hz with a resonance q value of 1.0 for all 3x speakers projecting through 3 dedicated loudspeakers respectfully. 2x 12db bell notches were also applied with slight variance depending on the frequency response of the respective loudspeaker.



4.1 Datasets Used

Datasets		
Training	Testing	Noise Injection
vctk	libri	musan
Audio Speech Actors	voxforge	wham
timit [4]		urbansound8k
libri		soundclas
		wass

GENERATED DATASETS:

Lets denote the generated datasets as:

1. `D_MIX`
2. `D_LIBRI`
3. `A_MIX`
4. `A_LIBRI`
5. `T_VOXFORGE`

D for digital as described in the first approach.

A for analog as described in the second approach.

T for test, which was recorded using the second approach.

D_MIX and A_MIX contain voices from multiple datasets and D_Libri and A_Libri only contain voices from the Libri-Clean dataset and T_Voxforge only contains voices from the voxforge dataset.

Generated Datasets (contains source files from)					
Voice or Noise Dataset	D_MIX	D_LIBRI	A_MIX	A_LIBRI	T_VOXFORGE
vctk [6]	x		x		
AudioSpeechActors [1]	x		x		
timit [4]	x		x		
libri [2]	x	x	x	x	
voxforge [7]					x
musan [3]	x	x	x	x	x
wham! [?]	x	x	x	x	x
urbansound8k [5]	x	x	x	x	x
soundclas	x	x	x	x	x
wass [10]	x	x	x	x	x

5 Experiments & Results

The experiment consists of training 4 models, 1 model for each generated dataset and then benchmarking them by calculating cosine-similarity, dynamic time warp and si-snr between each inference result and ground truth. What we hope to see here is a significant difference between the generated datasets "D" and "A".

5.1 Training & Evaluation

Each dataset was trained on a model with the same hyper parameters (see appendix 2.3 for details). Some datasets took longer to train, since they varied in size.

Training				
Training & Evaluation Metric	D_MIX	D_LIBRI	A_MIX	A_LIBRI
Epoches	30/100	14/100	45/100	32/100
Best Loss (SI-SDR)	30.4511	23.0362	-3.2646	-0.4677
Training Batches	14000	63000	2000	1000
Evaluation Batches	2200	10000	400	200

5.2 Testing & Analysis

5.2.1 The Analysis Algorithm

The analysis algorithm is as follows:

1. Select the datasets to be used.
2. Split the source files into respective folders and DAW Projects.
3. Generate the Datasets
4. Convert the audio to appropriate format.
5. Create appropriate training and development scp files.
6. Train the model(s).
7. Conduct Inference on the models.
8. Create appropriate evaluation scp files.
9. Evaluate the model(s).
10. Plot the collected results.

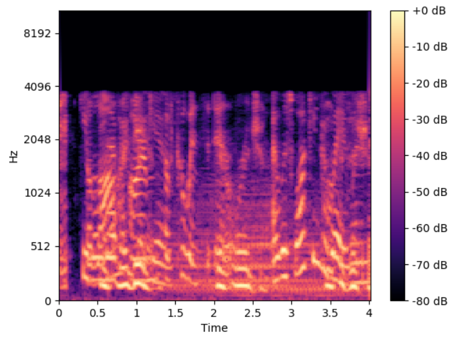
The Cosine-similarity, Dynamic Time Warp and SI-SNR (Scale Invariant - Signal To Noise Ratio) was calculated between the model inference results and ground truths respectfully. Inference and evaluation was conducted on the voxforge [7] dataset for 234 mixtures. Here is a table with the averages for each metric.

Difference Metrics				
Metric (Average)	D_MIX	D_LIBRA	A_MIX	A_LIBRA
Cosine-Similarity	0.2669	0.08136	0.1440	0.43402
Dynamic Time Warp	33.3772	36.9084	25.4251	25.9138
SI-SNR	-19.5565	-18.3159	-21.1566	-14.2978

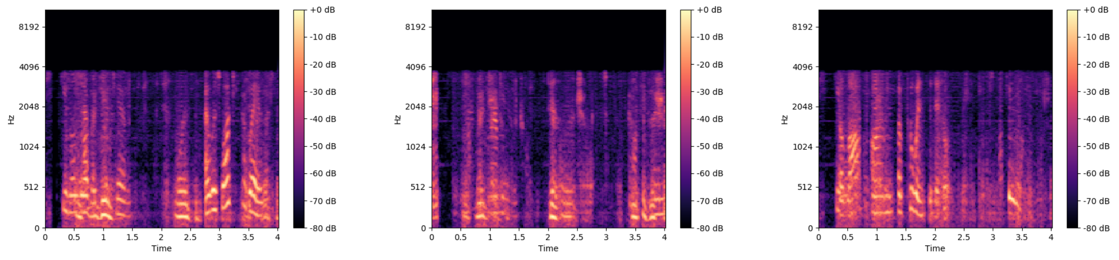
See appendix for plots.

As we can see here there is a significant favorable difference between the A_LIBRA dataset for Cosine-Similarity and the SI-SNR compared to the rest. The Dynamic Time Warp Values are also lower for the recorded datasets. However, since the A_MIX has the lowest SI-SNR and a low cos-sine similarity average, we can not be certain that the proposed method makes a significant improvement. More research and improvement on the testing method is needed. Moreover,subjective listening sessions were conducted as well as the generation of some visual log-scale mel spectrum plots to see if separation was occurring aurally and visually.

Composition_14



Separated speakers $\longrightarrow$ spk_1, spk_2, spk_3



6 Conclusion

This paper presents a dataset creation methodology focused on improving 3 speaker separation for Conv-Tasnet neural net model. The core of the design relies on the use of publicly available voice datasets, a digital audio workstation and various audio related recording hardware. Digital voice records are pre-processed, mixed together by projecting each voice through a separate loudspeaker and converted to reduce training time and memory requirements. This data is then passed to the Conv-Tasnet architecture. Finally the cosine similarity, dynamic time warp and si-snr across all inferences and ground truths are calculated. The results indicate that by using this technique with further investigation, may be able to improve training the Conv-TasNet architecture over using pure digital 3 speaker dataset compositions.

7 Future Works

Perform an improved comparative analysis using the exact same source files and injected noise. This would greatly improve the accuracy for determining the difference between dataset generation approaches.

References

[1] Ravdess. URL `https://smartlaboratory.org/ravdess/`.

[2] URL `http://www.openslr.org/12/`.

[3] musan dataset. URL `https://www.openslr.org/17/`.

[4] Timit acoustic-phonetic continuous speech corpus. URL `https://catalog.ldc.upenn.edu/LDC93S1`.

[5] Urbansound8k. URL `https://urbansounddataset.weebly.com/urbansound8k.html`.

[6] Vctk. URL `https://datashare.is.ed.ac.uk/handle/10283/2950`.

[7] URL `http://www.voxforge.org/home/downloads`.

[8] Funcwj. funcwj/conv-tasnet, Nov 2019. URL `https://github.com/funcwj/conv-tasnet`.

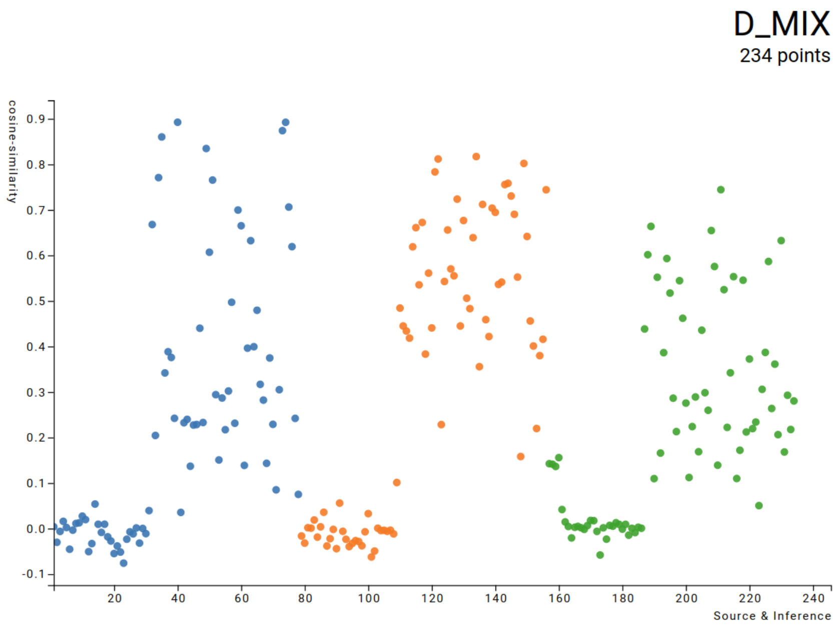
[9] Y. Luo and N. Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1256–1266, 2019. doi: 10.1109/taslp.2019.2915167.

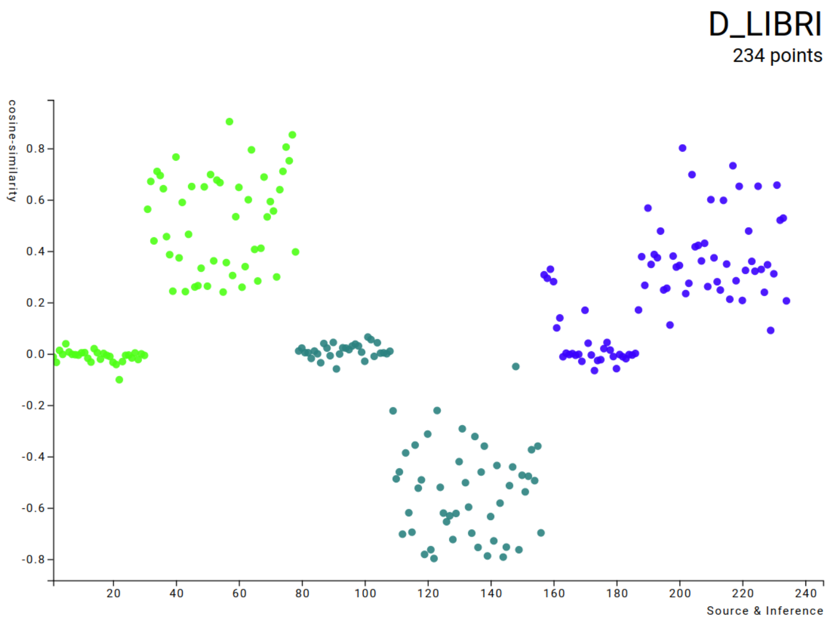
[10] Mitmedialab. mitmedialab/wetlandaviansourceseparation. URL `https://github.com/mitmedialab/WetlandAvianSourceSeparation`.

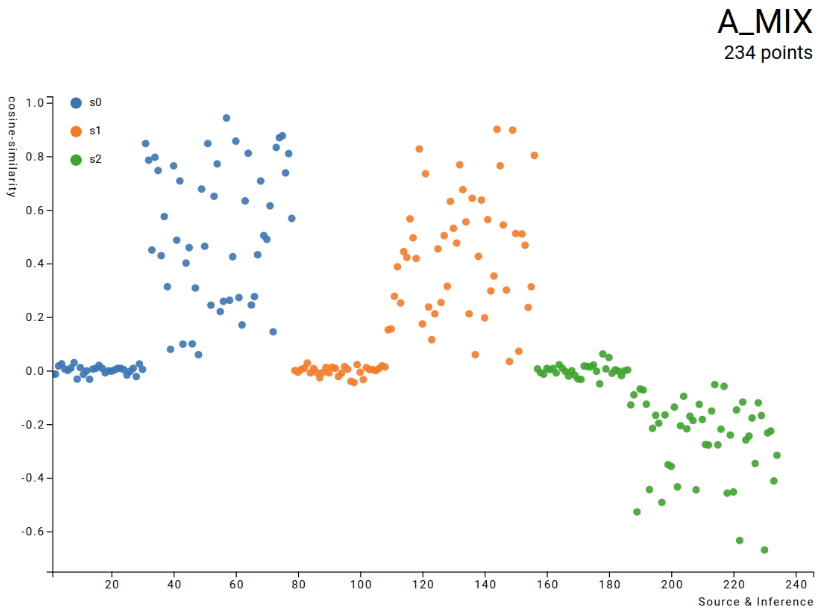
8 Appendix

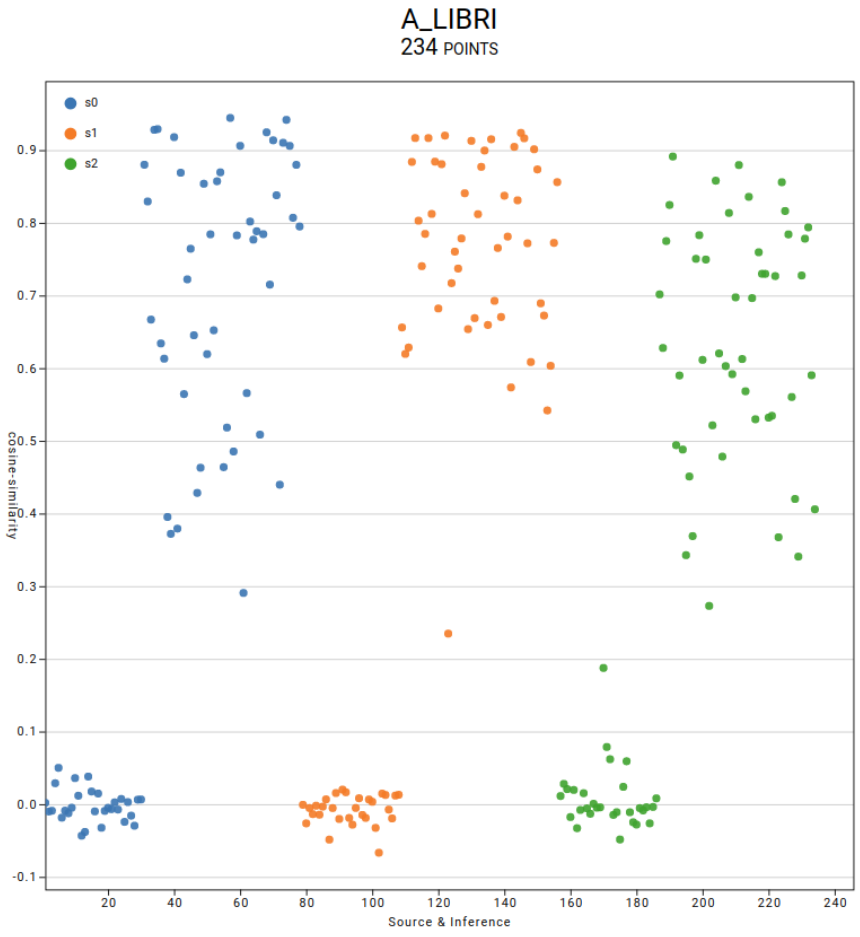
8.1 Plots

8.1.1 Cosine-Similarity Plots

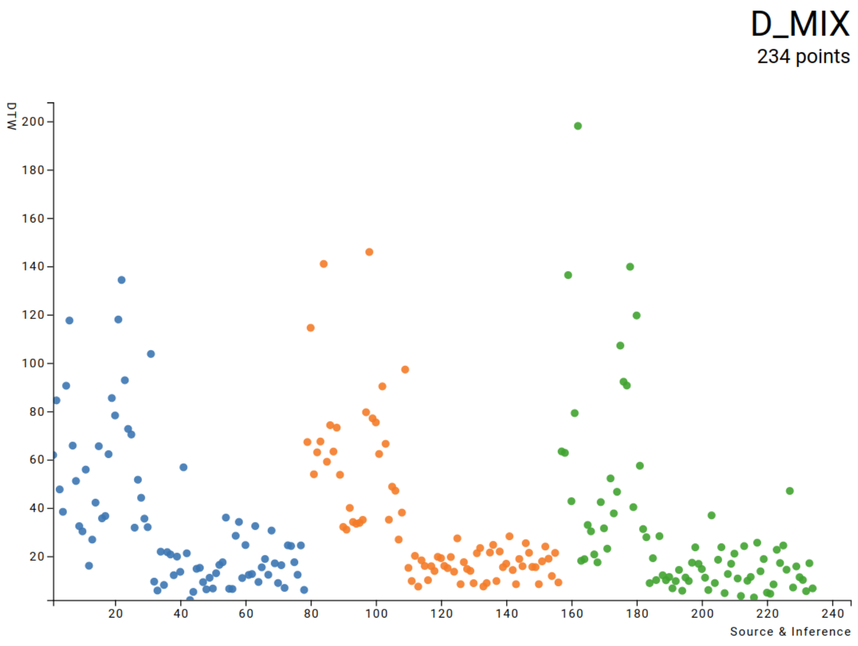


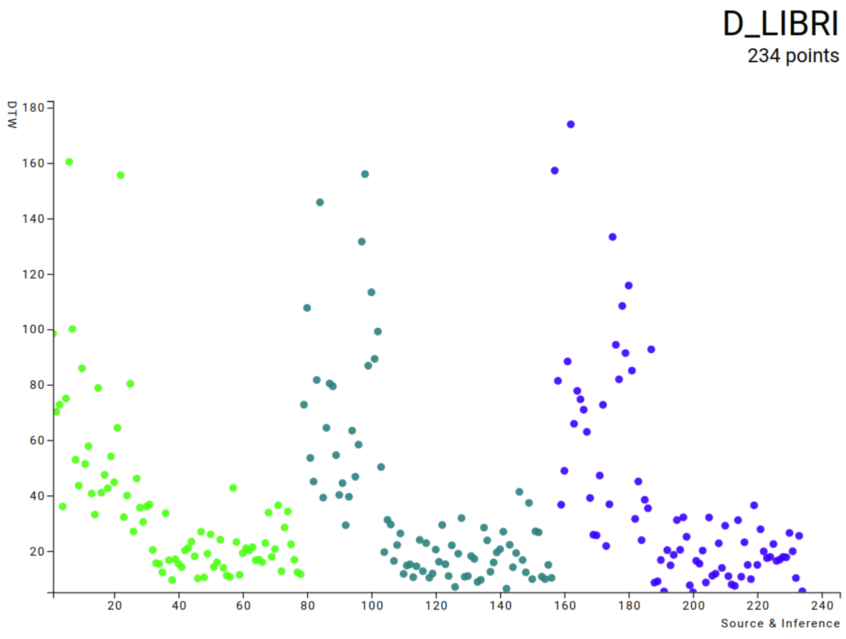


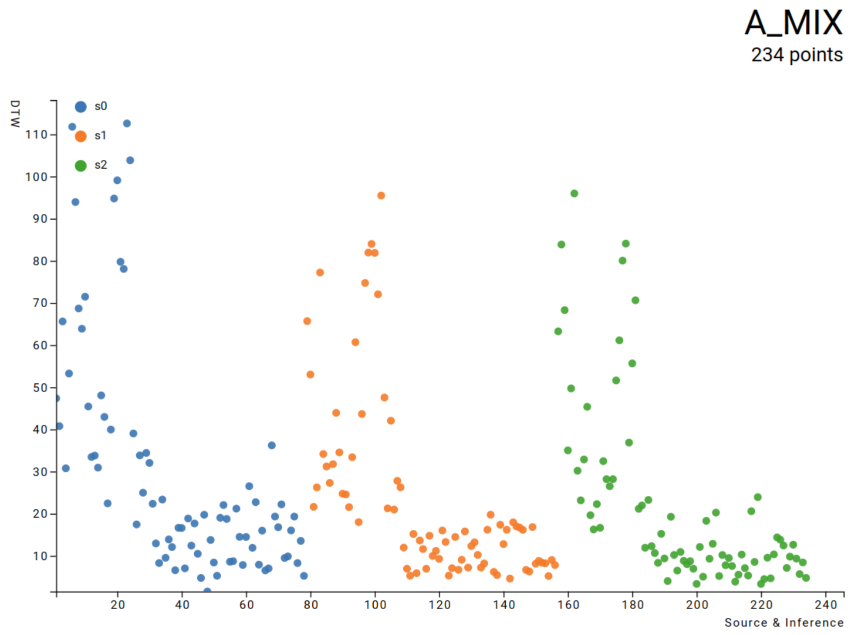

A_MIX
234 points
cosine-similarity
s0
s1
s2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
20
40
60
80
100
120
140
160
180
200
220
240
Source & Inference

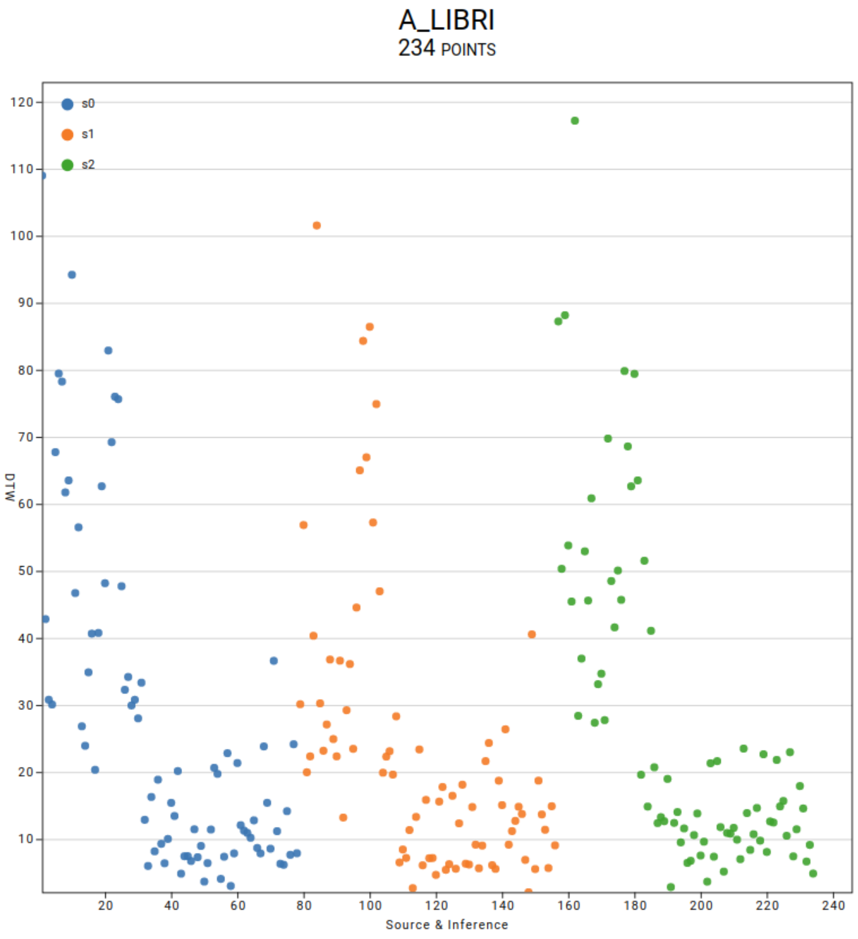

A_LIBRI
234 POINTS
cosine-similarity
s0
s1
s2
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
20
40
60
80
100
120
140
160
180
200
220
240
Source & Inference

8.1.2 Dynamic Time Warp (DTW) Plots

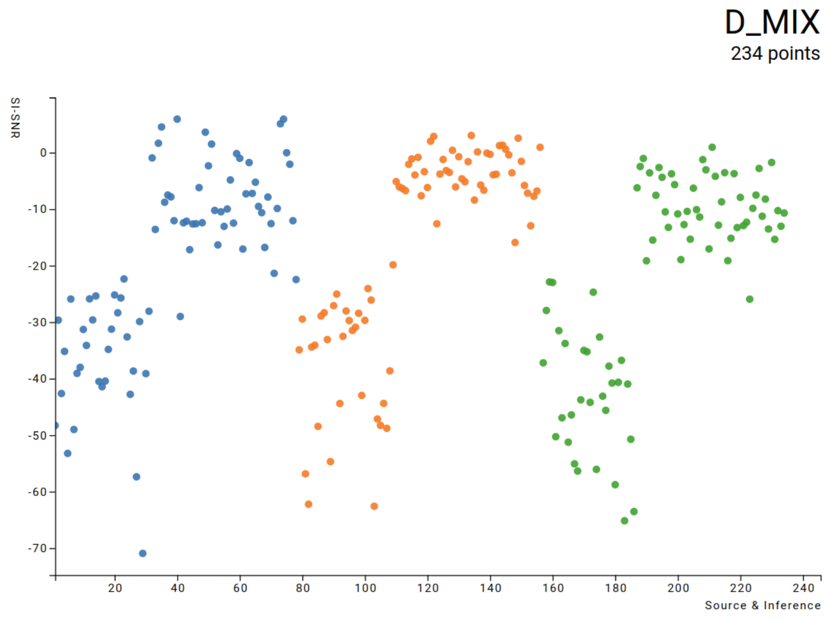


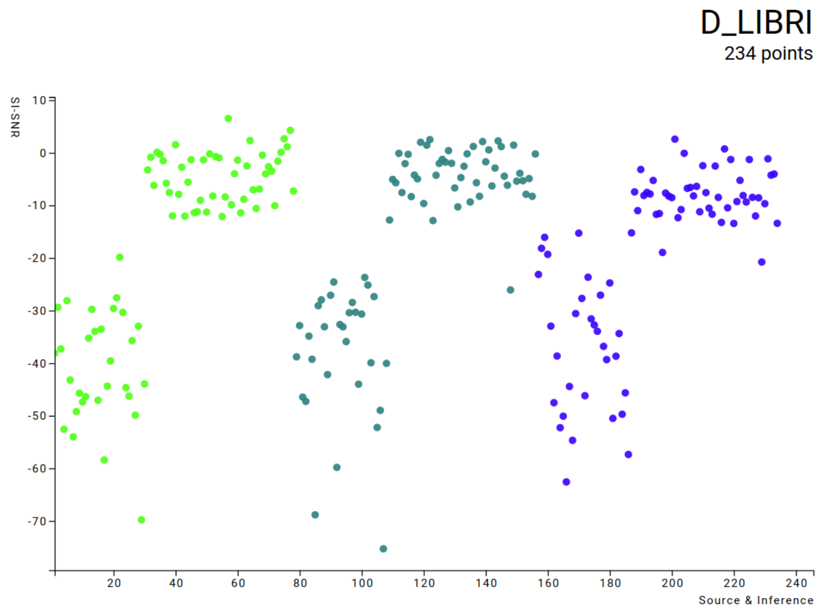


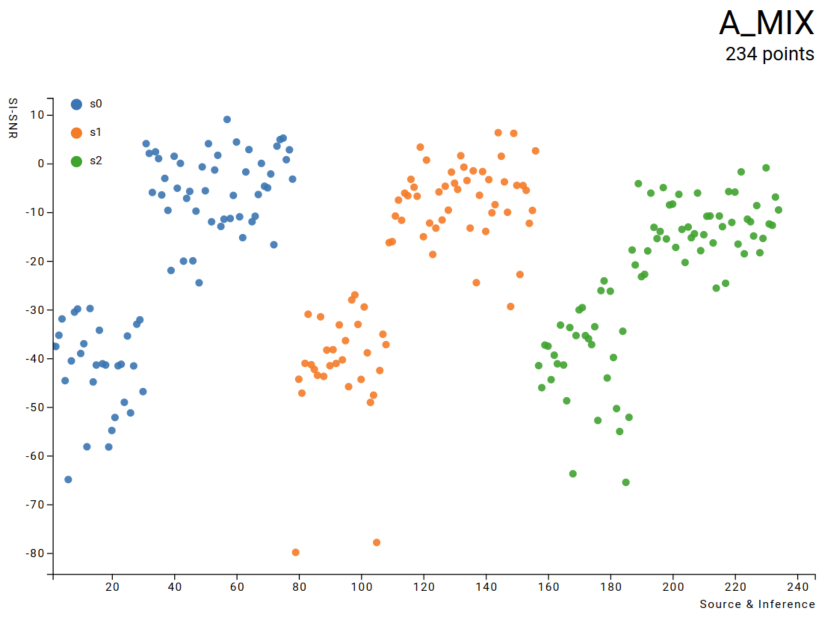

A_MIX
234 points
DTW
s0
s1
s2
Source & Inference

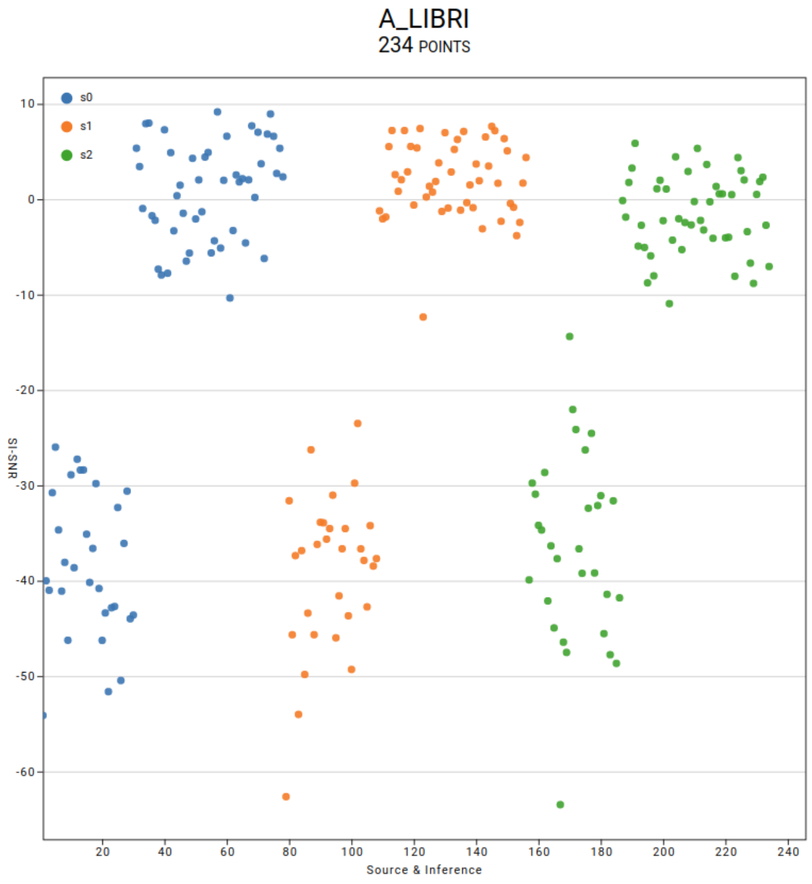

A_LIBRI
234 POINTS
DTW
s0
s1
s2
Source & Inference

8.1.3 SI-SNR Plots






A_MIX
234 points
s0
s1
s2
SI-SNR
10
0
-10
-20
-30
-40
-50
-60
-70
-80
20
40
60
80
100
120
140
160
180
200
220
240
Source & Inference


A_LIBRI
234 POINTS
s0
s1
s2
SI-SNR
10
0
-10
-20
-30
-40
-50
-60
20
40
60
80
100
120
140
160
180
200
220
240
Source & Inference

8.2 Model Details (same for all models)

```
{
    "L": 20,
    "N": 256,
    "X": 8,
    "R": 4,
    "B": 256,
    "H": 512,
    "P": 3,
    "norm": "BN",
    "num_spks": 3,
    "non_linear": "relu"
}
```

```
    ConvTasNet(
  (encoder_1d): Conv1D(1, 256, kernel_size=(20,), stride=(10,))
  (ln): ChannelWiseLayerNorm((256,), eps=1e-05, elementwise_affine=True)

  (proj): Conv1D(256, 256, kernel_size=(1,), stride=(1,))
  (repeats): Sequential(
    (0): Sequential(
      (0): Conv1DBlock(
        (conv1x1): Conv1D(256, 512, kernel_size=(1,), stride=(1,))
        (prelu1): PReLU(num_parameters=1)
        (lnorm1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
        (dconv): Conv1d(512, 512, kernel_size=(3,), stride=(1,), padding=(1,)
        , groups=512)
        (prelu2): PReLU(num_parameters=1)
        (lnorm2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
        (sconv): Conv1d(512, 256, kernel_size=(1,), stride=(1,))
      )

 ...
      (7): Conv1DBlock(
        (conv1x1): Conv1D(256, 512, kernel_size=(1,), stride=(1,))
        (prelu1): PReLU(num_parameters=1)
        (lnorm1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
        (dconv): Conv1d(512, 512, kernel_size=(3,), stride=(1,), padding=(128,),
        dilation=(128,), groups=512)
        (prelu2): PReLU(num_parameters=1)
        (lnorm2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
        (sconv): Conv1d(512, 256, kernel_size=(1,), stride=(1,))
      )
    )
  )
  (mask): Conv1D(256, 768, kernel_size=(1,), stride=(1,))
  (decoder_1d): ConvTrans1D(256, 1, kernel_size=(20,), stride=(10,))
)
```

8.3 Training Details

```
{
    "optimizer": "adam",
    "optimizer_kwargs": {
        "lr": 0.001,
        "weight_decay": 1e-05
    },
    "min_lr": 1e-08,
    "patience": 2,
    "factor": 0.5,
    "logging_period": 200
}
```

8.3.1 D_MIX

```
2020-03-17 15:58:45 [.../trainer.py:179 - INFO ] Create optimizer adam:
{'lr': 0.001, 'weight_decay': 1e-05}
2020-03-17 15:58:45 [.../trainer.py:148 - INFO ] Model summary:
...
2020-03-17 15:58:45 [.../trainer.py:150 - INFO ] Loading model to
GPUs:(0,), #param: 8.82M
2020-03-17 15:58:46 [.../trainer.py:205 - INFO ] Set eval mode...
...
2020-03-18 16:53:55 [.../trainer.py:66 - INFO ] Processed 1000 batches(loss = +30.13)...
2020-03-18 16:54:09 [.../trainer.py:66 - INFO ] Processed 1200 batches(loss = +33.67)...
```

```
2020-03-18 16:54:21 [.../trainer.py:66 - INFO ] Processed 1400 batches(loss = +30.35)...
2020-03-18 16:54:35 [.../trainer.py:66 - INFO ] Processed 1600 batches(loss = +29.30)...
2020-03-18 16:54:47 [.../trainer.py:66 - INFO ] Processed 1800 batches(loss = +26.81)...
2020-03-18 16:55:03 [.../trainer.py:66 - INFO ] Processed 2000 batches(loss = +29.46)...
2020-03-18 16:55:21 [.../trainer.py:66 - INFO ] Processed 2200 batches(loss = +31.97)...
2020-03-18 16:55:26 [.../trainer.py:73 - INFO ] Loss on {:d} batches: ...
2020-03-18 16:55:26 [.../trainer.py:252 - INFO ] Loss(time/N, lr=6.250e-05) - Epoch 30:
train = +28.4658(46.77m/14062) | dev = +30.4917(2.75m/2250) | no impr, best = 30.4511
2020-03-18 16:55:27 [.../trainer.py:262 - INFO ] Stop training cause no
impr for 6 epochs
2020-03-18 16:55:27 [.../trainer.py:265 - INFO ] Training for 30/100
epoches done!
```

8.3.2 D_LIBRI

```
2020-03-24 13:25:47 [.../trainer.py:179 - INFO ] Create optimizer adam: {'lr': 0.001,
'weight_decay': 1e-05}
2020-03-24 13:25:47 [.../trainer.py:148 - INFO ]
...
2020-03-24 13:25:47 [.../trainer.py:150 - INFO ] Loading model to GPUs:(0,), #param:
8.82M
2020-03-24 13:25:48 [.../trainer.py:205 - INFO ] Set eval mode...
...
2020-03-26 17:55:55 [.../trainer.py:66 - INFO ] Processed 9400 batches(loss = +21.43)...
2020-03-26 17:56:08 [.../trainer.py:66 - INFO ] Processed 9600 batches(loss = +20.76)...
2020-03-26 17:56:22 [.../trainer.py:66 - INFO ] Processed 9800 batches(loss = +22.68)...
2020-03-26 17:56:32 [.../trainer.py:66 - INFO ] Processed 10000 batches(loss = +21.14)..
2020-03-26 17:56:39 [.../trainer.py:73 - INFO ] Loss on {:d} batches: ...
2020-03-26 17:56:39 [.../trainer.py:252 - INFO ] Loss(time/N, lr=5.000e-04) - Epoch 14:
train = +21.4329(213.27m/63000) | dev = +23.0836(11.19m/10080) | no impr, best = 23.0362
2020-03-26 17:56:40 [.../trainer.py:262 - INFO ] Stop training cause no impr for 6
epochs
2020-03-26 17:56:40 [.../trainer.py:265 - INFO ] Training for 14/100 epoches done!
```

8.3.3 A_MIX

```
2020-03-29 18:04:17 [.../trainer.py:179 - INFO ] Create optimizer adam: {'lr': 0.001,
'weight_decay': 1e-05}
2020-03-29 18:04:17 [.../trainer.py:148 - INFO ]
...
```

```
2020-03-29 18:09:55 [.../trainer.py:150 - INFO ] Loading model to GPUs:(0,), #param:
8.82M
2020-03-29 18:10:34 [.../trainer.py:205 - INFO ] Set eval mode...
...
2020-03-29 23:50:26 [.../trainer.py:66 - INFO ] Processed 2000 batches(loss = -10.18)...
2020-03-29 23:50:36 [.../trainer.py:205 - INFO ] Set eval mode...
2020-03-29 23:50:47 [.../trainer.py:66 - INFO ] Processed 200 batches(loss = -3.21)...
2020-03-29 23:50:58 [.../trainer.py:66 - INFO ] Processed 400 batches(loss = -3.07)...
2020-03-29 23:51:04 [.../trainer.py:73 - INFO ] Loss on {:d} batches: ...
2020-03-29 23:51:04 [.../trainer.py:252 - INFO ] Loss(time/N, lr=1.250e-04) - Epoch 45:
train = -10.4459(6.76m/2052) | dev = -3.2288(0.46m/513) | no impr, best = -3.2646
2020-03-29 23:51:04 [.../trainer.py:262 - INFO ] Stop training cause no impr for 6 epoch
2020-03-29 23:51:04 [.../trainer.py:265 - INFO ] Training for 45/100 epoches done!
```

8.3.4 A_LIBRI

```
2020-04-05 17:49:21 [.../trainer.py:179 - INFO ] Create optimizer adam: {'lr': 0.001,
'weight_decay': 1e-05}
2020-04-05 17:49:21 [.../trainer.py:148 - INFO ]
...
2020-04-05 17:49:21 [.../trainer.py:150 - INFO ] Loading model to GPUs:(0,), #param:
8.82M
2020-04-05 17:49:21 [.../trainer.py:205 - INFO ] Set eval mode...
...
2020-04-05 19:42:32 [.../trainer.py:66 - INFO ] Processed 1000 batches(loss = -6.71)...
2020-04-05 19:42:32 [.../trainer.py:205 - INFO ] Set eval mode...
2020-04-05 19:42:44 [.../trainer.py:66 - INFO ] Processed 200 batches(loss = -0.28)...
2020-04-05 19:42:47 [.../trainer.py:73 - INFO ] Loss on {:d} batches: ...
2020-04-05 19:42:48 [.../trainer.py:252 - INFO ] Loss(time/N, lr=1.250e-04) - Epoch 32:
train = -6.7243(3.37m/1001) | dev = -0.2762(0.26m/250) | no impr, best = -0.4677
2020-04-05 19:42:48 [.../trainer.py:262 - INFO ] Stop training cause no impr for 6
epochs
2020-04-05 19:42:48 [.../trainer.py:265 - INFO ] Training for 32/100 epoches done!
```