Improving Sound Separation Using Sound Classification

Efthymios Tzinis\textsuperscript{1,2}, Scott Wisdom\textsuperscript{1}, John R. Hershey\textsuperscript{1}, Aren Jansen\textsuperscript{1}, Daniel P. W. Ellis\textsuperscript{1}

\textsuperscript{1}Google Research
\textsuperscript{2}University of Illinois at Urbana-Champaign

International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2020
Ideally we want to automatically separate all types of sounds
Ideally we want to automatically separate all types of sounds

Prior work: End-to-end universal sound separation [1]

- 10 dB SI-SDRi but still behind STFT oracle binary mask result of 16 dB
- Assuming that sound detection is easier than separation
  - What if we could detect the sources in a mixture?

Ideally we want to automatically separate all types of sounds

Prior work: End-to-end universal sound separation [1]
- 10 dB SI-SDRi but still behind STFT oracle binary mask result of 16 dB
- Assuming that sound detection is easier than separation
  - What if we could detect the sources in a mixture?

Potential pitfalls of end-to-end separation networks training:
- Can the neural network practically learn a good decomposition for all sounds of interest?
- Might not be the best way to utilize the high-level semantics of sounds
- A separation network might need a bit of guidance

Idea: Guiding source separation using semantic representations audio sources

1. A neural network performing source separation on a mixture of signals
Idea: Guiding source separation using semantic representations audio sources

1. A neural network performing source separation on a mixture of signals

2. Extract a high-level semantic representation for the input audio “conditional embedding”
1. A neural network performing source separation on a mixture of signals

2. **Extract a high-level semantic representation** for the input audio “conditional embedding”

3. **Guide/condition** the separation network using this embedding in order to improve its accuracy
Baseline Separation Network: (similar to ConvTasNet [2])

- **Analysis/Synthesis Basis:**
  - **Learnable:** 1D convolution/deconvolution layers
  - **Fixed:** STFT basis

Separation Network: Time-Dilated Convolution Network (TDCN++)

Baseline Separation Network: (similar to ConvTasNet [2])

- **Analysis/Synthesis Basis:**
  - **Learnable:** 1D convolution/deconvolution layers
  - **Fixed:** STFT basis

- **Separator:**
  - 32 1D Separable convolutional blocks
  - Residual connections from previous blocks

---

Separation Network: Time-Dilated Convolution Network (TDCN++)

**Baseline Separation Network:** (similar to ConvTasNet [2])
- **Analysis/Synthesis Basis:**
  - **Learnable:** 1D convolution/deconvolution layers
  - **Fixed:** STFT basis
- **Separator:**
  - 32 1D Separable convolutional blocks
  - Residual connections from previous blocks

**Loss:**
- **Permutation Invariant Signal to Noise Ratio (SNR)**

\[ \mathcal{L} = -SNR(s_{p^*}, \hat{s}) = -10 \log_{10} \frac{||s_{p^*}||^2}{||s_{p^*} - \hat{s}||^2} \]

Extract audio embeddings from a pre-trained sound classifier

Sound classifier:
- Event sound classifier (**527 classes**)
- Trained on **AudioSet**
- **MobileNet** for audio

How good are these embeddings?
- The sound classifier has also been trained using **mixtures** of sounds
Type of frame-wise conditional embeddings

- Embeddings of the source signals
  - An angry horse
  - Insect hissing
Type of frame-wise conditional embeddings

- Embeddings of the source signals
  - An angry horse
  - Insect hissing

- Embedding of the mixture signal:
  - Not always enclosing the semantic information of all the sources
Type of frame-wise conditional embeddings

- Embeddings of the **source** signals
  - An angry horse
  - Insect hissing

- Embedding of the **mixture** signal:
  - Not always enclosing the semantic information of all the sources

- Soft OR embedding:
  - The probability that one or more sources is active
Integrating semantic information in TDCN++

Integrate at i-th layer of a TDCN++:

![Diagram of TDCN++ integrating semantic information at the i-th layer.](image)
Integrating semantic information in TDCN++

Integrate at i-th layer of a TDCN++:

1. **Resample** the embedding in time
Integrating semantic information in TDCN++

Integrate at i-th layer of a TDCN++:

1. **Resample** the embedding in time
2. Apply a **sigmoid** on the embedding vector
Integrating semantic information in TDCN++

Integrate at i-th layer of a TDCN++:

1. **Resample** the embedding in time
2. Apply a **sigmoid** on the embedding vector
3. **Reduce** channels dimensions
Integrating semantic information in TDCN++

Integrate at i-th layer of a TDCN++:

1. **Resample** the embedding in time
2. **Apply a sigmoid** on the embedding vector
3. **Reduce** channels dimensions
4. **Global normalization**
Integrating semantic information in TDCN++

Integrate at i-th layer of a TDCN++:

1. **Resample** the embedding in time
2. Apply a **sigmoid** on the embedding vector
3. **Reduce** channels dimensions
4. **Global normalization**
5. **Combine** with activations:
   a. Concatenate $u_{i-1} = [V_{in}, y_{i-1}] \in \mathbb{R}^{W \times (B+B')}$
   b. Gating $u_{i-1} = V_{in} \odot y_{i-1} \in \mathbb{R}^{W \times B}$
TDCN++ with pre-trained embeddings

Baseline experiment:
- Using the embedding only from the input mixture

\[
\mathcal{L} = -SNR(s_p^*, \hat{s}) = -10 \log_{10} \frac{\|s_p^*\|^2}{\|s_p^* - \hat{s}\|^2}
\]
**Oracle experiment** with “all” embeddings:
- Concatenate the embeddings for mixture and all the sources
  - This is a **measure of upper bound of the performance improvement** we can get from the integration of the semantic information

**Oracle experiment** with soft-OR embedding
TDCN++ with fine-tuned embeddings

Problem: The pre-trained mixture embedding
- Is not fine-tuned for the task for separation
- Embeddings are trained on different data and task

\[
\mathcal{L} = -SNR(s_p^*, \hat{s}) = -10 \log_{10} \frac{\|s_p^*\|^2}{\|s_p^* - \hat{s}\|^2}
\]
Problem: The pre-trained mixture embedding
- Is **not fine-tuned for the task** for separation
- Embeddings are trained on **different data and task**

Idea: Refining the embeddings before conditioning
- **Fine-tuning** the last layers of the sound classifier

\[
\mathcal{L} = -SNR(s^*, \hat{s}) = -10 \log_{10} \frac{||s^*||^2}{||s^* - \hat{s}||^2}
\]
Problem: The pre-trained mixture embedding
● Is not fine-tuned for the task for separation
● Embeddings are trained on different data and task

Idea: Refining the embeddings before conditioning
● Fine-tuning the last layers of the sound classifier
● End-to-end source separation:
  ○ The loss remains the same as before

\[ \mathcal{L} = -SNR(s_p^*, \hat{s}) = -10 \log_{10} \frac{\|s_p^*\|^2}{\|s_p^* - \hat{s}\|^2} \]
First estimate the sources and then extract the conditional embeddings.

The Premise:
- **Using embeddings from clean sources** might lead to better separation performance [SPOILER ALERT]
The Premise:

- Using embeddings from clean sources might lead to better separation performance [SPOILER ALERT]

Idea: Extending end-to-end architecture for getting “all” embeddings

1. Try to separate the sources first
The Premise:

- **Using embeddings from clean sources** might lead to better separation performance [SPOILER ALERT]

Idea: Extending end-to-end architecture for getting “all” embeddings

1. **Try to separate the sources first**
2. **Use the first estimates of the sources in order to extract embeddings corresponding to the clean sources**
Iterative separation and refinement of embeddings (iTDCN++)

Architecture main points:
1. **Estimate the separated sources** and then extract the embeddings for both the estimates and the input mixture
Iterative separation and refinement of embeddings (iTDCN++)

Architecture main points:

1. **Estimate the separated sources** and then extract the embeddings for both the estimates and the input mixture
2. Use the estimates and the embeddings for **making better the final separation**
Architecture main points:
1. **Estimate the separated sources** and then extract the embeddings for both the estimates and the input mixture
2. Use the estimates and the embeddings for **making better the final separation**

Source separation losses:
1. First separation estimation: $\mathcal{L}_{sep}^{(1)} = -SNR(s_{p^*}, \hat{s}^{(1)})$
Architecture main points:
1. **Estimate the separated sources** and then extract the embeddings for both the estimates and the input mixture
2. Use the estimates and the embeddings for making better the final separation

Source separation losses:
1. First separation estimation: \( \mathcal{L}_{sep}^{(1)} = -SNR\left(s_{p^{*}}, \hat{s}^{(1)}\right) \)
2. Final separation estimation: \( \mathcal{L}_{sep}^{(2)} = -SNR\left(s_{p^{*}}, \hat{s}^{(2)}\right) \)
Guided Iterative separation and fine-tuned embeddings

Idea: Use the “ideal” embeddings as targets

Embeddings Losses: sigmoid cross-entropy (SCE)
Idea: Use the “ideal” embeddings as targets

Embeddings Losses: sigmoid cross-entropy (SCE)

- Making the mixture embedding look like the soft OR embedding:
  \[ \mathcal{L}_{emb}^{(1)} = SCE \left( \hat{V}_{m}^{(1)}, \hat{V}_{m}^{(1)} \right) \]
Guided Iterative separation and fine-tuned embeddings

Idea: Use the “ideal” embeddings as targets

Embeddings Losses: sigmoid cross-entropy (SCE)

- Making the mixture embedding look like the soft OR embedding:
  \[ L_{emb}^{(1)} = SCE\left(\hat{V}_{or}^p, \hat{V}_m^{(1)}\right) \]

- Making the sources embeddings look like the target ones:
  \[ L_{emb}^{(2)} = SCE\left(\hat{V}_{or}^p, \hat{V}_m^{(2)}\right) + SCE\left(\hat{V}_s^p, \hat{V}_s^{(2)}\right) \]
Experiments on Universal Sound Separation

Task:
- 2-source separation
Experiments on Universal Sound Separation

Task:
- 2-source separation

Prosound Dataset:
- Wide variety of sound classes
  - (animal calls, musical instruments, speech, artificial sounds, etc.)
  - 3 seconds clips sampled at 16kHz
- Train/Val/Test splits:
  - 11.7 hours training mixtures
  - 3.2 hours validation mixtures
  - 1.7 hours test mixtures
Experiments on Universal Sound Separation

Task:
- 2-source separation

Prosound Dataset:
- Wide variety of sound classes
  - (animal calls, musical instruments, speech, artificial sounds, etc.)
  - 3 seconds clips sampled at 16kHz
- Train/Val/Test splits:
  - 11.7 hours training mixtures
  - 3.2 hours validation mixtures
  - 1.7 hours test mixtures

Evaluation Metric:
- Permutation-invariant scale-invariant signal-to-distortion ratio improvement (SI-SDRi)
## Performance (SI-SDR improvement in dB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Embeddings</th>
<th>STFT</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Val.</td>
<td>Test</td>
</tr>
<tr>
<td><strong>Baselines</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDCN++ with no embeddings [8]</td>
<td>-</td>
<td>9.9</td>
<td>9.1</td>
</tr>
<tr>
<td>iTDCN++ with no embeddings [8]</td>
<td>-</td>
<td>10.6</td>
<td>9.8</td>
</tr>
<tr>
<td><strong>Proposed</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pretrained embeddings &amp; TDCN++</td>
<td>mixture</td>
<td>10.3</td>
<td>9.4</td>
</tr>
<tr>
<td>Fine-tuned embeddings &amp; TDCN++</td>
<td>mixture</td>
<td>10.2</td>
<td>9.4</td>
</tr>
<tr>
<td>Guided fine-tuned embeddings &amp; TDCN++</td>
<td>mixture</td>
<td>10.3</td>
<td>9.4</td>
</tr>
<tr>
<td>Pretrained embeddings &amp; iTDCN++</td>
<td>all</td>
<td>10.8</td>
<td>9.9</td>
</tr>
<tr>
<td>Fine-tuned embeddings &amp; iTDCN++</td>
<td>all</td>
<td>11.1</td>
<td>10.1</td>
</tr>
<tr>
<td>Guided fine-tuned embeddings &amp; iTDCN++</td>
<td>all</td>
<td>11.1</td>
<td>10.2</td>
</tr>
<tr>
<td><strong>Oracles</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pretrained embeddings &amp; TDCN++</td>
<td>all</td>
<td>11.3</td>
<td>10.6</td>
</tr>
<tr>
<td>STFT binary mask</td>
<td>soft-OR</td>
<td>11.4</td>
<td>10.6</td>
</tr>
</tbody>
</table>

1. **Consistent performance improvement when we use embeddings for source separation**
   a. Improvement also when simple pre-trained embeddings are used
### Performance (SI-SDR improvement in dB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Embeddings</th>
<th>STFT</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Val.</td>
<td>Test</td>
</tr>
<tr>
<td><strong>Baselines</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDCN++ with no embeddings [8]</td>
<td>-</td>
<td>9.9</td>
<td>9.1</td>
</tr>
<tr>
<td>iTDCN++ with no embeddings [8]</td>
<td>-</td>
<td>10.6</td>
<td>9.8</td>
</tr>
<tr>
<td><strong>Proposed</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pretrained embeddings &amp; TDCN++</td>
<td>mixture</td>
<td>10.3</td>
<td>9.4</td>
</tr>
<tr>
<td>Fine-tuned embeddings &amp; TDCN++</td>
<td>mixture ✔</td>
<td>10.2</td>
<td>9.4</td>
</tr>
<tr>
<td>Guided fine-tuned embeddings &amp; TDCN++</td>
<td>mixture ✔</td>
<td>10.3</td>
<td>9.4</td>
</tr>
<tr>
<td>Pretrained embeddings &amp; iTDCN++</td>
<td>all</td>
<td>10.8</td>
<td>9.9</td>
</tr>
<tr>
<td>Fine-tuned embeddings &amp; iTDCN++</td>
<td>all ✔</td>
<td>11.1</td>
<td>10.1</td>
</tr>
<tr>
<td>Guided fine-tuned embeddings &amp; iTDCN++</td>
<td>all ✔</td>
<td>11.1</td>
<td>10.2</td>
</tr>
<tr>
<td><strong>Oracles</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pretrained embeddings &amp; TDCN++</td>
<td>all</td>
<td>11.3</td>
<td>10.6</td>
</tr>
<tr>
<td>STFT binary mask</td>
<td>-</td>
<td>16.8</td>
<td>16.2</td>
</tr>
</tbody>
</table>

1. **Consistent performance improvement when we use embeddings for source separation**
   a. Improvement also when simple pre-trained embeddings are used
   b. Improvement also with the simpler end-to-end approach
### Performance (SI-SDR improvement in dB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Embeddings Type</th>
<th>Embeddings Fine-tuning</th>
<th>STFT Val.</th>
<th>STFT Test</th>
<th>Learned Val.</th>
<th>Learned Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baselines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDCN++ with no embeddings [8]</td>
<td>-</td>
<td>-</td>
<td>9.9</td>
<td>9.1</td>
<td>9.1</td>
<td>8.5</td>
</tr>
<tr>
<td>iTDCN++ with no embeddings [8]</td>
<td>-</td>
<td>-</td>
<td>10.6</td>
<td><strong>9.8</strong></td>
<td>9.3</td>
<td><strong>8.7</strong></td>
</tr>
<tr>
<td>Proposed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pretrained embeddings &amp; TDCN++</td>
<td>mixture</td>
<td>-</td>
<td>10.3</td>
<td>9.4</td>
<td>9.4</td>
<td>8.6</td>
</tr>
<tr>
<td>Fine-tuned embeddings &amp; TDCN++</td>
<td>mixture</td>
<td>✓</td>
<td>10.2</td>
<td>9.4</td>
<td>9.3</td>
<td>8.5</td>
</tr>
<tr>
<td>Guided fine-tuned embeddings &amp; TDCN++</td>
<td>mixture</td>
<td>✓</td>
<td>10.3</td>
<td>9.4</td>
<td>9.4</td>
<td>8.6</td>
</tr>
<tr>
<td>Pretrained embeddings &amp; iTDCN++</td>
<td>all</td>
<td>-</td>
<td>10.8</td>
<td>9.9</td>
<td>9.9</td>
<td>9.0</td>
</tr>
<tr>
<td>Fine-tuned embeddings &amp; iTDCN++</td>
<td>all</td>
<td>✓</td>
<td><strong>11.1</strong></td>
<td>10.1</td>
<td><strong>10.1</strong></td>
<td><strong>9.2</strong></td>
</tr>
<tr>
<td>Guided fine-tuned embeddings &amp; iTDCN++</td>
<td>all</td>
<td>✓</td>
<td><strong>11.1</strong></td>
<td><strong>10.2</strong></td>
<td>10.0</td>
<td>9.1</td>
</tr>
<tr>
<td>Oracles</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pretrained embeddings &amp; TDCN++</td>
<td>all</td>
<td>-</td>
<td>11.3</td>
<td>10.6</td>
<td>11.0</td>
<td>10.2</td>
</tr>
<tr>
<td>STFT binary mask</td>
<td>soft-OR</td>
<td>-</td>
<td>11.4</td>
<td>10.6</td>
<td>10.7</td>
<td>10.1</td>
</tr>
</tbody>
</table>

1. **Consistent performance improvement when we use embeddings for source separation**
   a. Improvement also when simple pre-trained embeddings are used
   b. Improvement also with the simpler end-to-end approach

2. Improvement over iTDCN++ **for the non-oracle case**: 0.4 dB (STFT basis) & 0.5 dB (Learnable basis)
### Performance (SI-SDR improvement in dB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Embeddings Type</th>
<th>Fine-tuning</th>
<th>STFT Val.</th>
<th>STFT Test</th>
<th>Learned Val.</th>
<th>Learned Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baselines</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDCN++ with no embeddings [8]</td>
<td>mixture</td>
<td></td>
<td>10.6</td>
<td>9.8</td>
<td>9.3</td>
<td>8.7</td>
</tr>
<tr>
<td>iTDCN++ with no embeddings [8]</td>
<td>all</td>
<td></td>
<td>9.9</td>
<td>9.1</td>
<td>9.1</td>
<td>8.5</td>
</tr>
<tr>
<td><strong>Proposed</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pretrained embeddings &amp; TDCN++</td>
<td>mixture</td>
<td></td>
<td>10.3</td>
<td>9.4</td>
<td>9.4</td>
<td>8.6</td>
</tr>
<tr>
<td>Fine-tuned embeddings &amp; TDCN++</td>
<td>all</td>
<td></td>
<td>11.1</td>
<td>10.1</td>
<td>10.1</td>
<td>9.2</td>
</tr>
<tr>
<td>Guided fine-tuned embeddings &amp; TDCN++</td>
<td>mixture</td>
<td></td>
<td>10.3</td>
<td>9.4</td>
<td>9.4</td>
<td>8.6</td>
</tr>
<tr>
<td><strong>Oracles</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pretrained embeddings &amp; TDCN++</td>
<td>all</td>
<td></td>
<td>11.3</td>
<td>10.6</td>
<td>11.0</td>
<td>10.2</td>
</tr>
<tr>
<td>STFT binary mask</td>
<td>soft-OR</td>
<td></td>
<td>11.4</td>
<td>10.6</td>
<td>10.7</td>
<td>10.1</td>
</tr>
</tbody>
</table>

1. **Consistent performance improvement when we use embeddings for source separation**
   a. Improvement also when simple pre-trained embeddings are used
   b. Improvement also with the simpler end-to-end approach
2. Improvement over iTDCN++ for the non-oracle case: **0.4 dB (STFT basis) & 0.5 dB (Learnable basis)**
3. Improvement over iTDCN++ for the oracle case: **0.8 dB (STFT basis) & 1.5 dB (Learnable basis)**
Conclusions & Future Work

Proposed

A new way to integrate semantic information of audio in order to perform higher quality universal sound separation.
Conclusions & Future Work

**Proposed**

A new way to integrate semantic information of audio in order to perform higher quality universal sound separation.

**Explored**

Trained and evaluated >1000 models with different parameter configurations. Variable ways of conditioning separation networks for better source separation.
Conclusions & Future Work

**Proposed**
A new way to integrate semantic information of audio in order to perform higher quality universal sound separation.

**Explored**
Trained and evaluated >1000 models with different parameter configurations. Variable ways of conditioning separation networks for better source separation.

**Results**
Our iterative approach achieves an improvement of 0.5 dB (learnable basis) and 0.4 dB (STFT basis) in SI-SDR over the baseline iterative model having no embeddings.
Conclusions & Future Work

Proposed
A new way to integrate semantic information of audio in order to perform higher quality universal sound separation.

Explored
Trained and evaluated >1000 models with different parameter configurations. Variable ways of conditioning separation networks for better source separation.

Results
Our iterative approach achieves an improvement of 0.5 dB (learnable basis) and 0.4 dB (STFT basis) in SI-SDR over the baseline iterative model having no embeddings.

Future
Check whether separated sounds help sound classification (there is DCASE 2020 Task 4 using the new Free Universal Sound Separation (FUSS) dataset that explores this task). Source separation with an unknown number of sources.
Thank you all!
Waiting to see you at the Q&A session!

Efthymios Tzinis