Improving Sound Separation Using Sound Classification

Efthymios Tzinis[1,2], Scott Wisdom[1], John R. Hershey[1], Aren Jansen[1], Daniel P. W. Ellis[1]
[1]Google Research
[2]University of Illinois at Urbana-Champaign

International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2020

Google AI

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**?

[1] Ilya Kavalerov, Scott Wisdom, Hakan Erdogan, Brian Patton, Kevin Wilson, Jonathan Le Roux, and John R Hershey, "Universal sound separation," Proc. WASPAA, 2019, pp. 175–179.

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**?

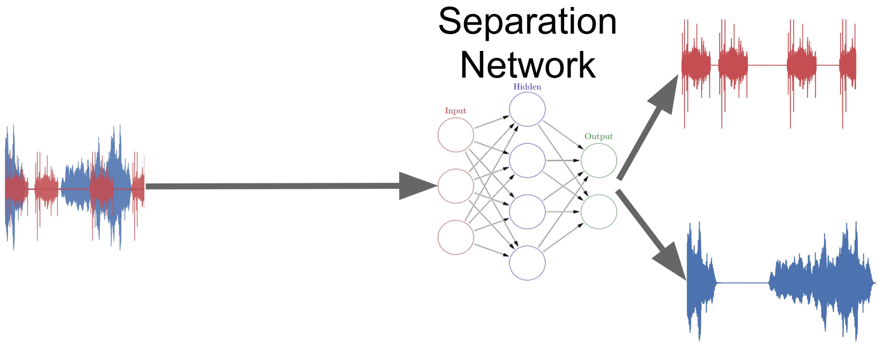
Potential pitfalls of end-to-end separation networks training:

- 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**

[1] Ilya Kavalerov, Scott Wisdom, Hakan Erdogan, Brian Patton, Kevin Wilson, Jonathan Le Roux, and John R Hershey, "Universal sound separation," Proc. WASPAA, 2019, pp. 175–179.

Idea: Guiding source separation using semantic representations audio sources

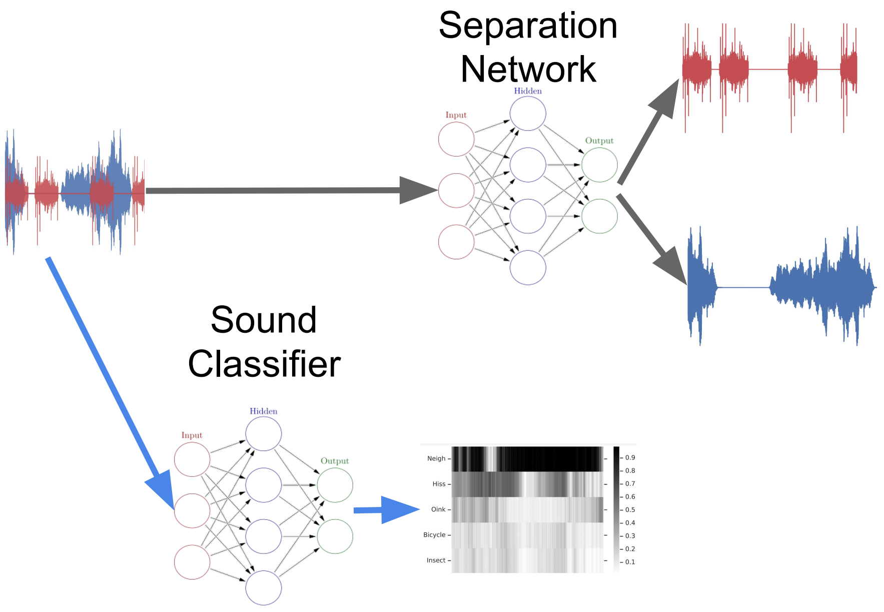
1. A neural network performing source separation on a mixture of signals



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"

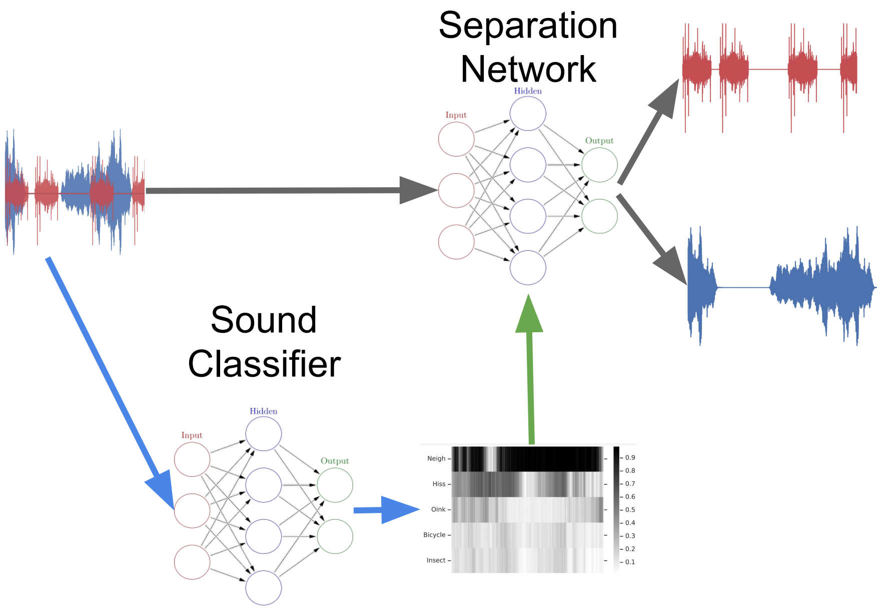


Idea: Guiding source separation using semantic representations audio sources

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"

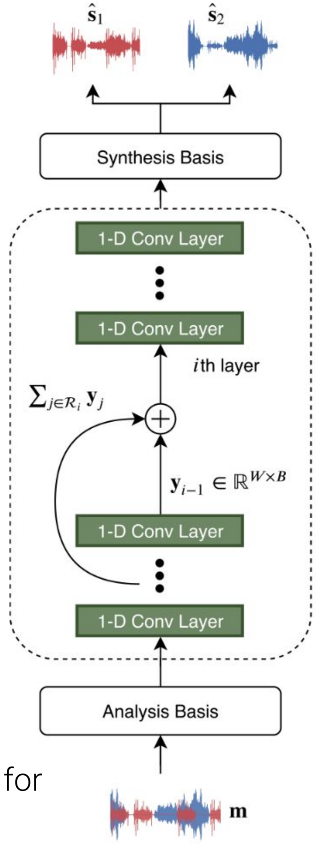
3. **Guide/condition** the separation network using this embedding in order to improve its accuracy



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

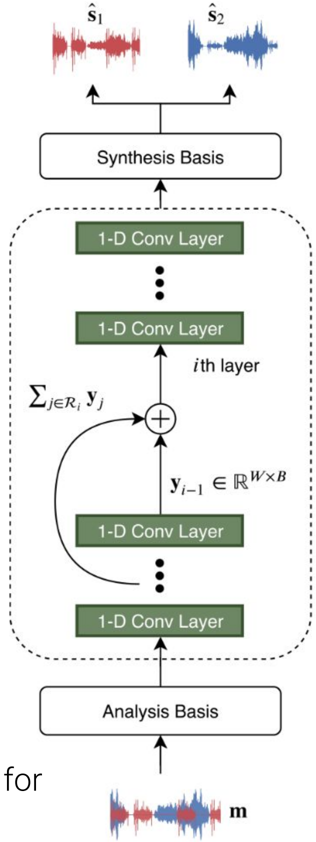


[2] Yi Luo and Nima Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2019.

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



[2] Yi Luo and Nima Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2019.

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

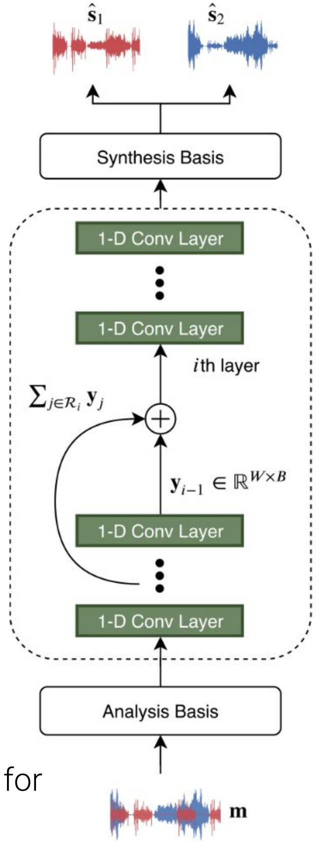
Baseline Separation Network: (similar to ConvTasNet [2])

- **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)**

$$\mathcal{L} = -SNR(\mathbf{s}_{p^*}, \hat{\mathbf{s}}) = -10 \log_{10} \frac{||\mathbf{s}_{p^*}||^2}{||\mathbf{s}_{p^*} - \hat{\mathbf{s}}||^2}$$



[2] Yi Luo and Nima Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2019.

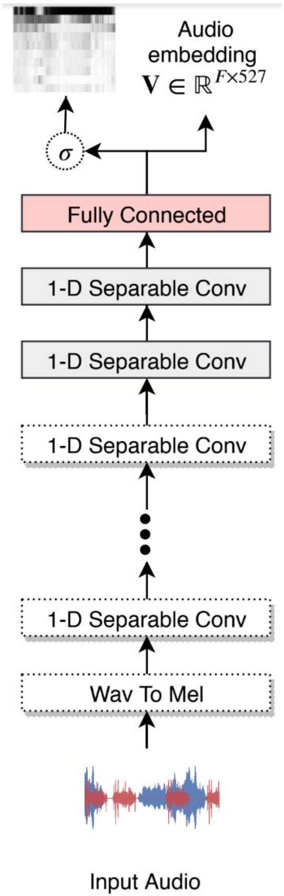
Extract audio embeddings from a pre-trained sound classifier

Sound classifier:

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

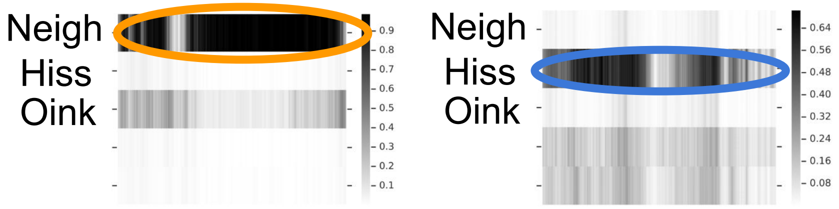
How good are these embeddings?

- The sound classifier has also been trained using **mixtures** of sounds



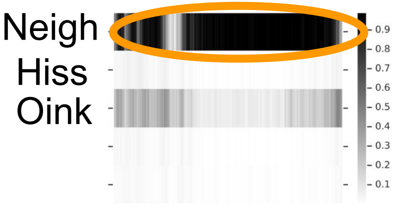
Type of frame-wise conditional embeddings

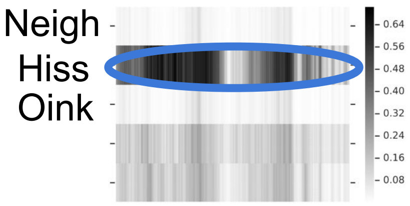
- Embeddings of the **source** signals
  - An angry horse
  - Insect hissing



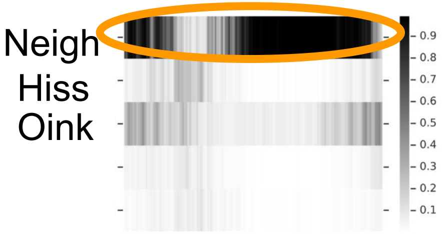
Type of frame-wise conditional embeddings

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





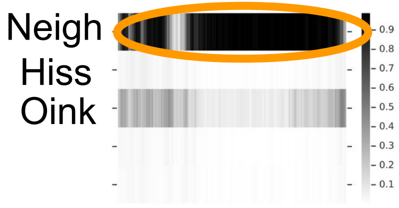
- Embedding of the **mixture** signal:
  - **Not always enclosing the semantic information of all the sources**

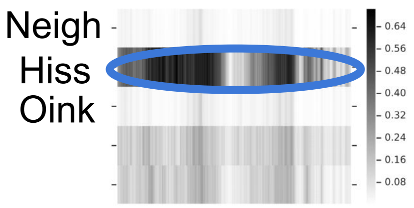


??

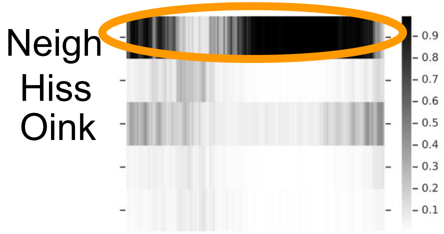
Type of frame-wise conditional embeddings

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



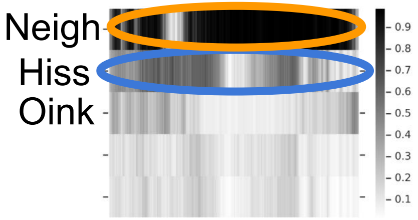


- Embedding of the **mixture** signal:
  - **Not always enclosing the semantic information of all the sources**



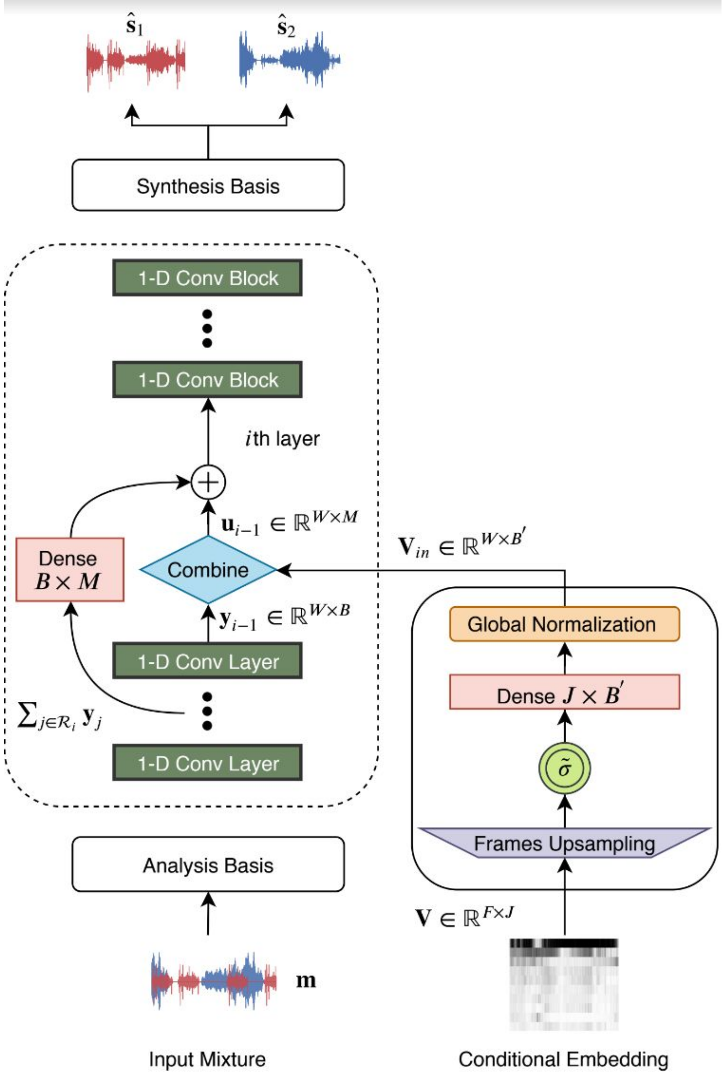
??

- **Soft OR** embedding:
  - The probability that one or more sources is active



Integrating semantic information in TDCN++

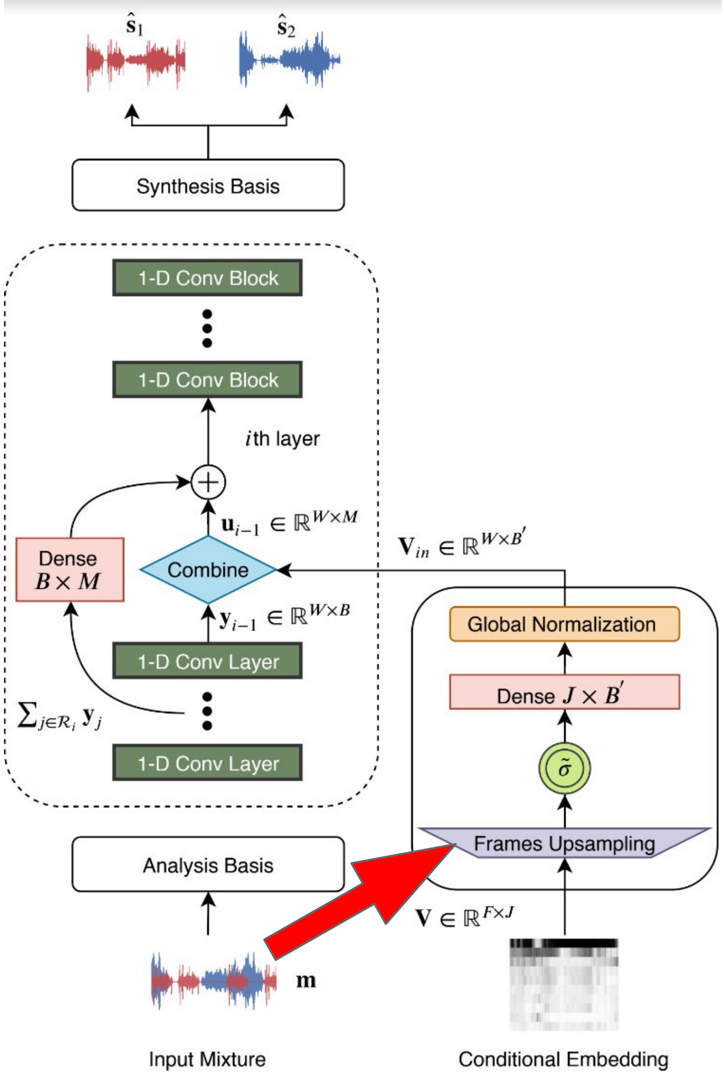
Integrate at i-th layer of a TDCN++ :



Integrating semantic information in TDCN++

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

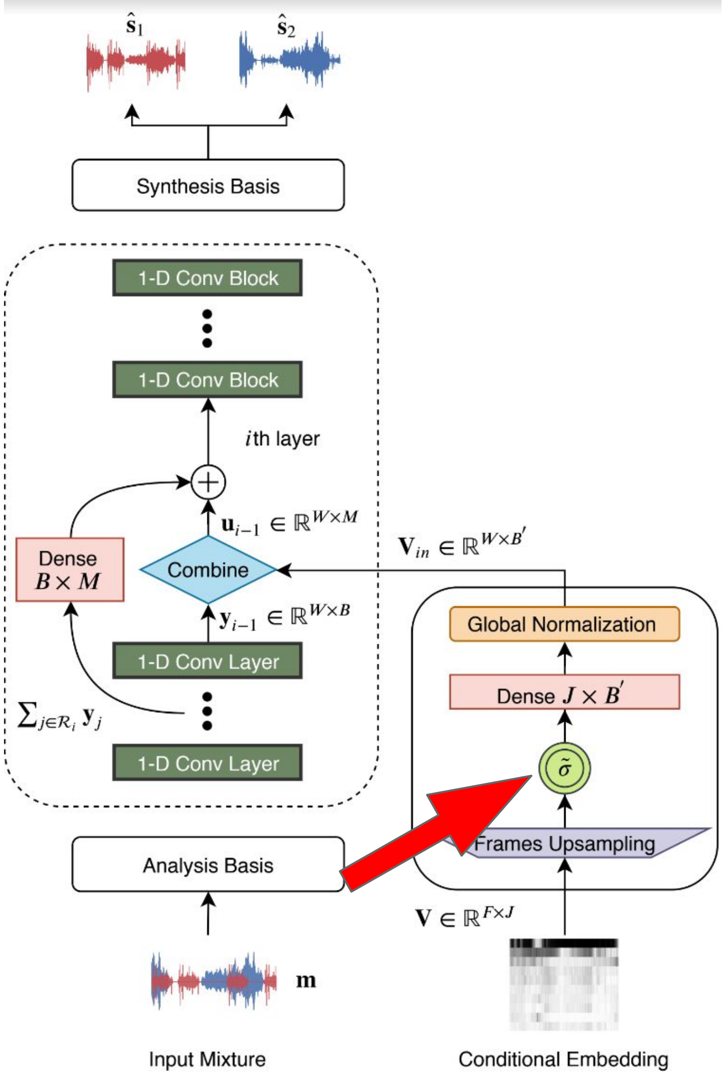
1. **Resample** the embedding in time



Integrating semantic information in TDCN++

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

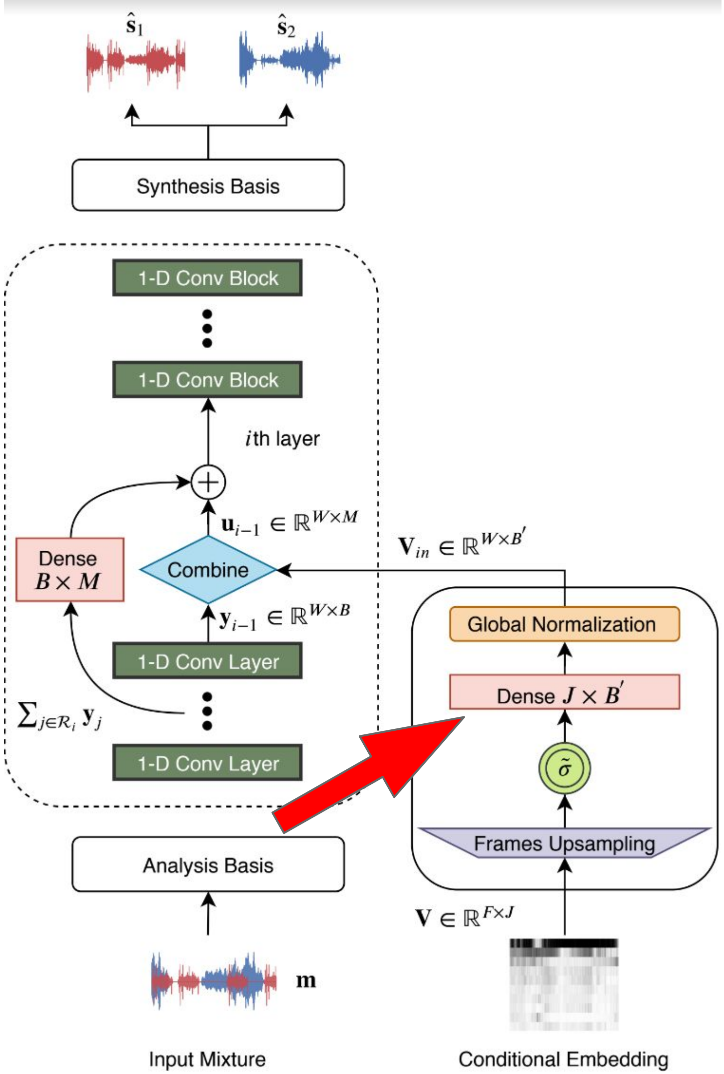
1. **Resample** the embedding in time
2. Apply a **sigmoid** on the embedding vector



Integrating semantic information in TDCN++

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

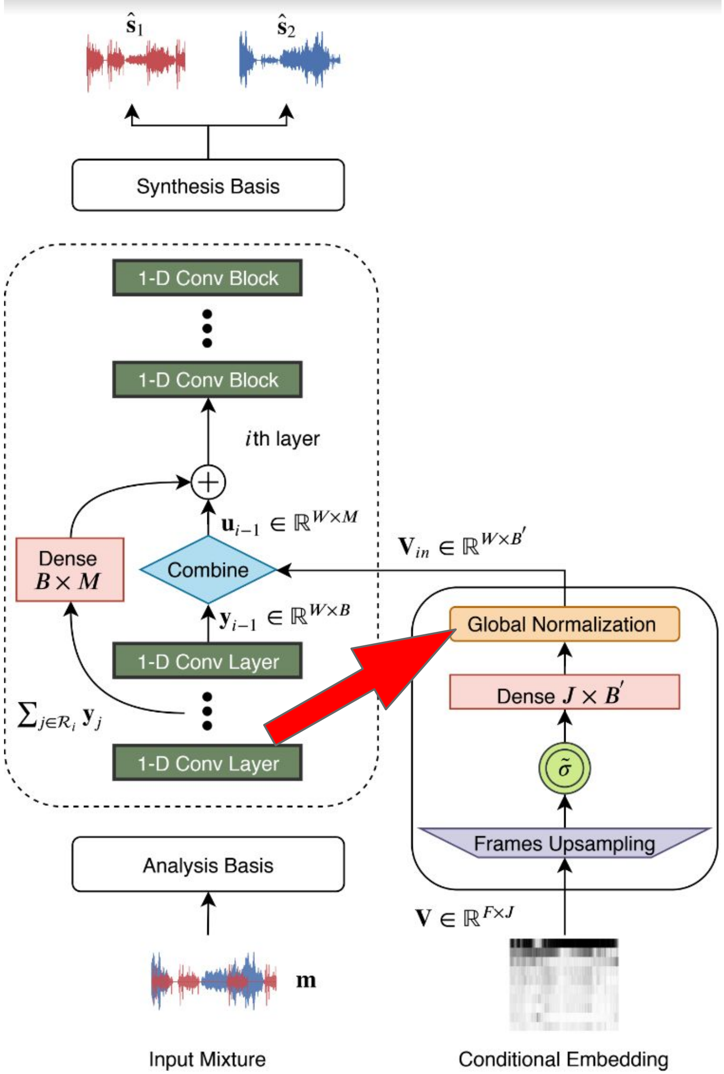
1. **Resample** the embedding in time
2. Apply a **sigmoid** on the embedding vector
3. **Reduce** channels dimensions



Integrating semantic information in TDCN++

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

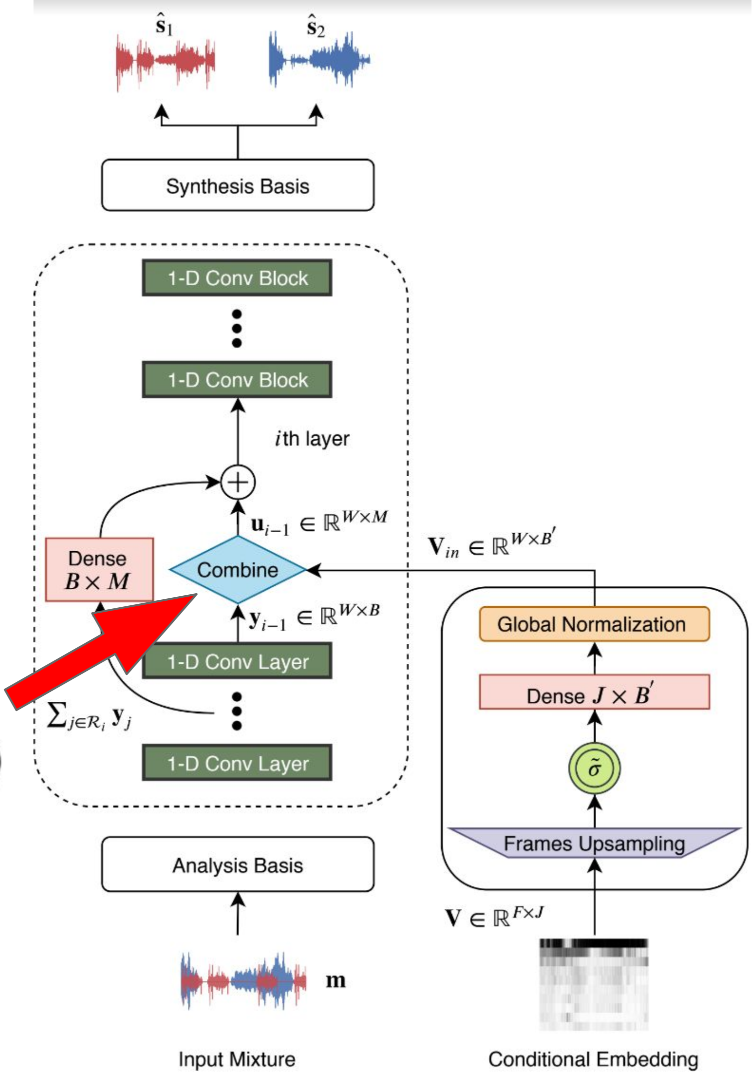
1. **Resample** the embedding in time
2. Apply a **sigmoid** on the embedding vector
3. **Reduce** channels dimensions
4. **Global normalization**



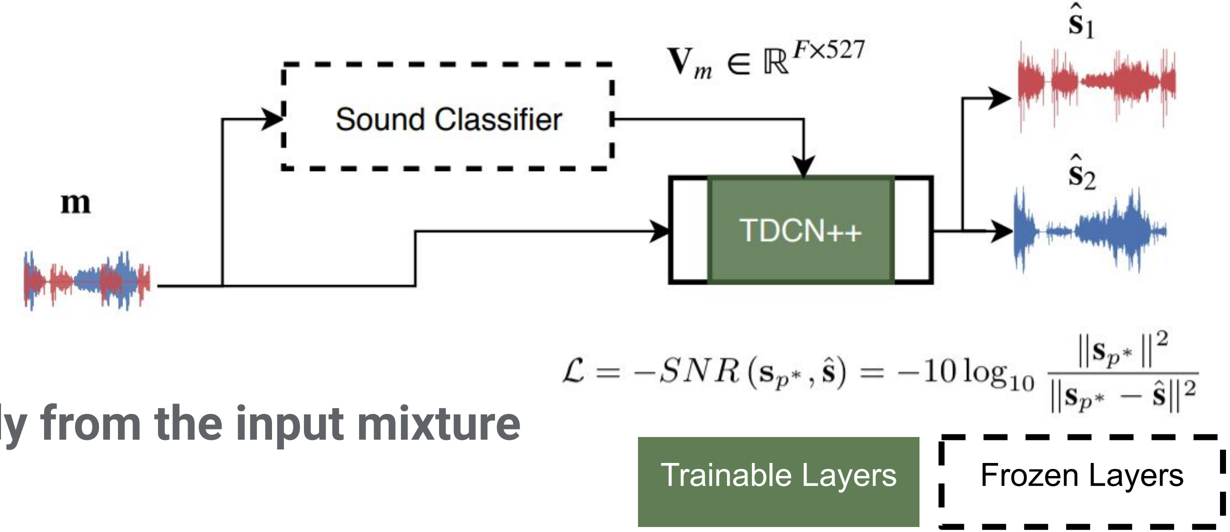
Integrating semantic information in TDCN++

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 $\mathbf{u}_{i-1} = [\mathbf{V}_{\text{in}}, \mathbf{y}_{i-1}] \in \mathbb{R}^{W \times (B+B')}$
   b. Gating $\mathbf{u}_{i-1} = \mathbf{V}_{\text{in}} \odot \mathbf{y}_{i-1} \in \mathbb{R}^{W \times B}$



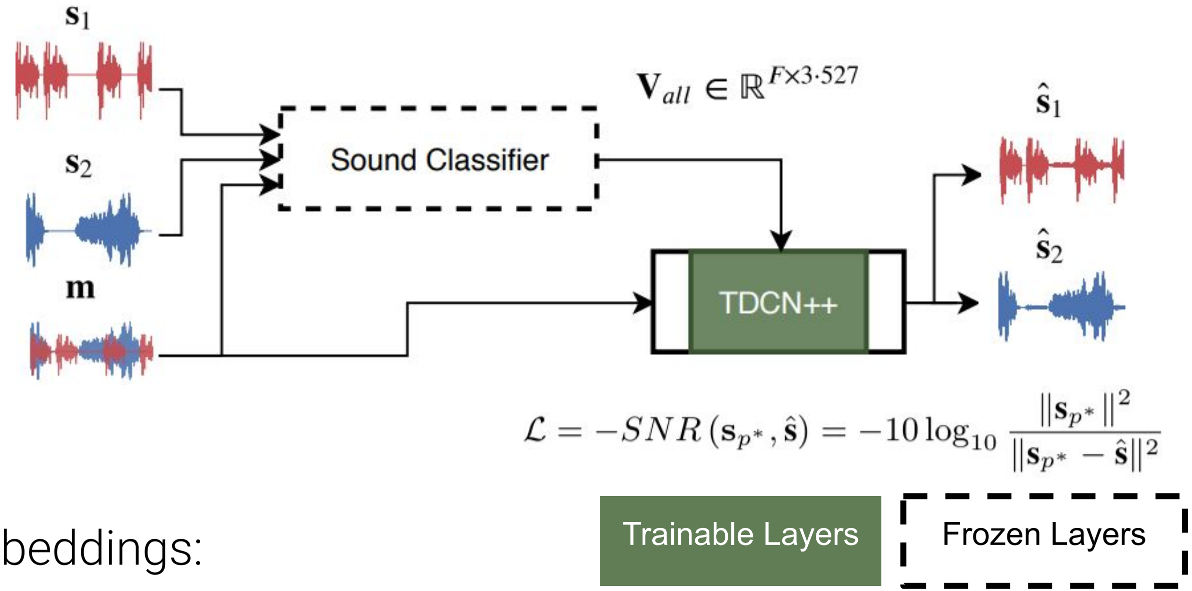
TDCN++ with pre-trained embeddings



Baseline experiment:

- Using the embedding **only from the input mixture**

TDCN++ with pre-trained embeddings



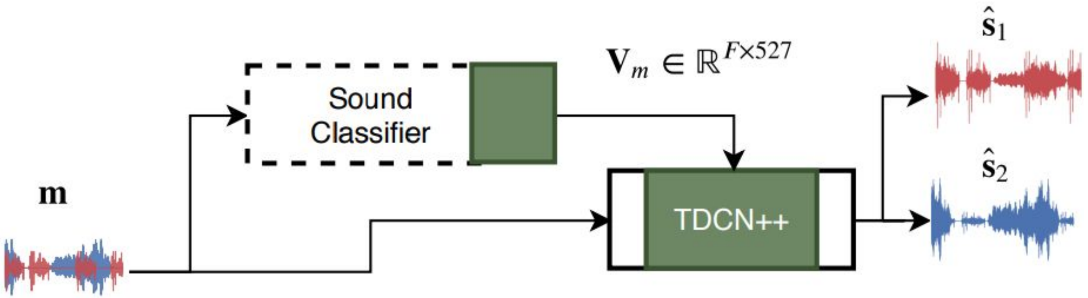
$$\mathcal{L} = -SNR(\mathbf{s}_{p^*}, \hat{\mathbf{s}}) = -10\log_{10}\frac{\|\mathbf{s}_{p^*}\|^2}{\|\mathbf{s}_{p^*} - \hat{\mathbf{s}}\|^2}$$

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

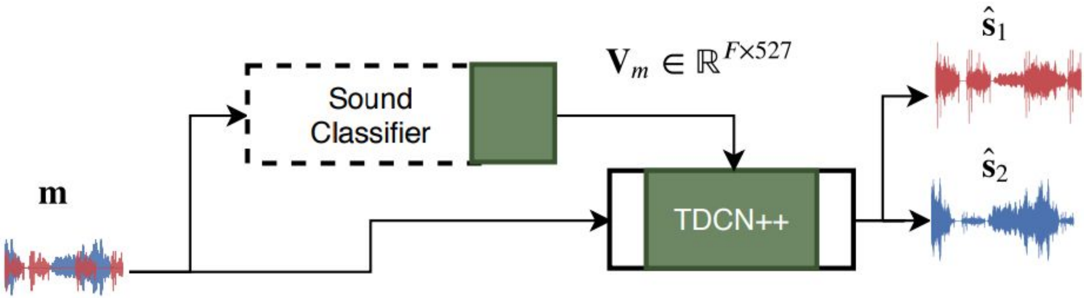
- Is **not fine-tuned for the task** for separation
- Embeddings are trained on **different data and task**

$$\mathcal{L} = -SNR(\mathbf{s}_{p^*}, \hat{\mathbf{s}}) = -10\log_{10}\frac{||\mathbf{s}_{p^*}||^2}{||\mathbf{s}_{p^*} - \hat{\mathbf{s}}||^2}$$

Trainable Layers

Frozen Layers

TDCN++ with fine-tuned embeddings



$$\mathcal{L} = -SNR(\mathbf{s}_{p^*}, \hat{\mathbf{s}}) = -10\log_{10}\frac{||\mathbf{s}_{p^*}||^2}{||\mathbf{s}_{p^*} - \hat{\mathbf{s}}||^2}$$

Trainable Layers | Frozen Layers

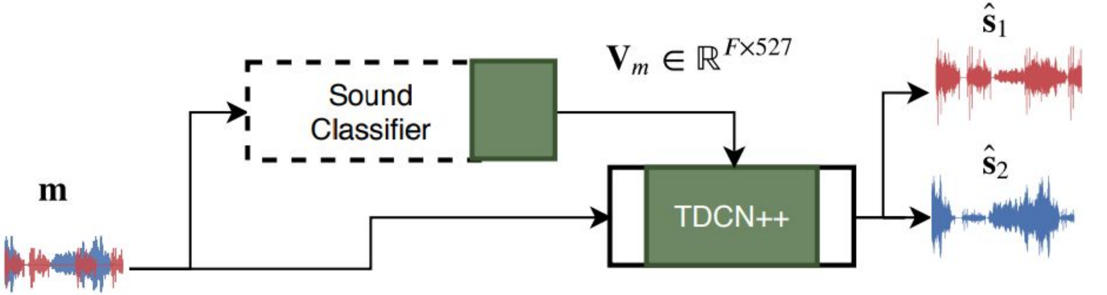
Problem: The pre-trained mixture embedding

- 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

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**

$$\mathcal{L} = -SNR(\mathbf{s}_{p^*}, \hat{\mathbf{s}}) = -10\log_{10}\frac{||\mathbf{s}_{p^*}||^2}{||\mathbf{s}_{p^*} - \hat{\mathbf{s}}||^2}$$

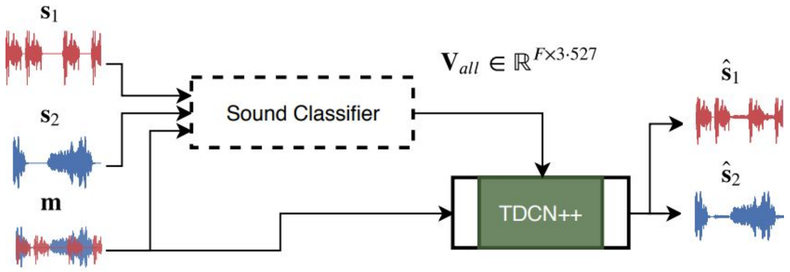
Trainable Layers

Frozen Layers

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**

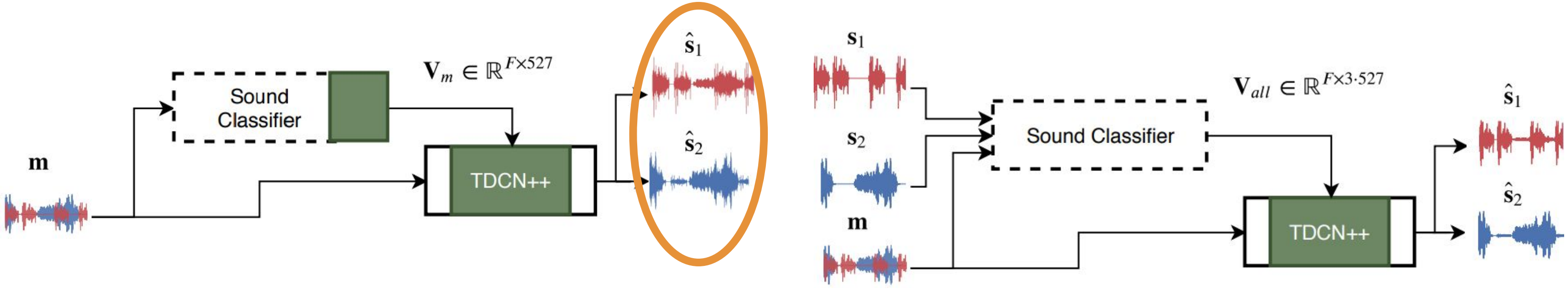
First estimate the sources and then extract the conditional embeddings



The Premise:

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

First estimate the sources and then extract the conditional embeddings



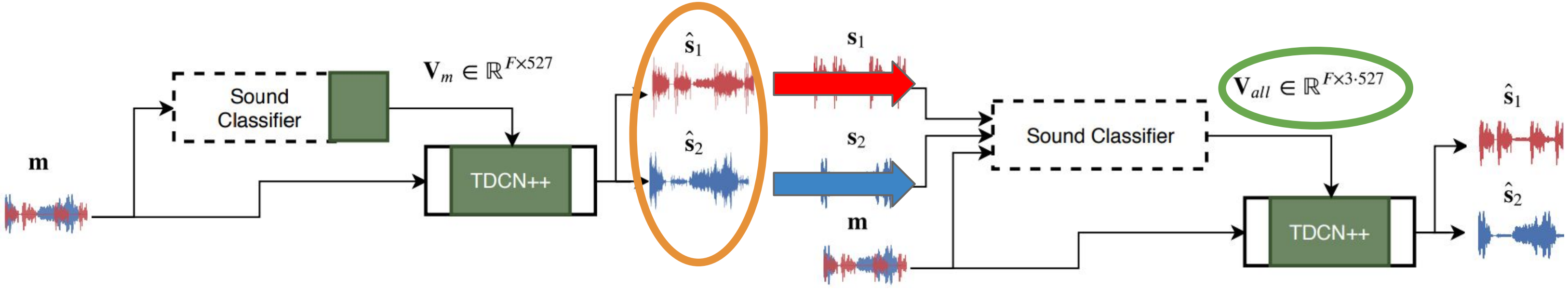
The Premise:

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

Idea: Extending end-to-end architecture for getting "all" embeddings

1. **Try to separate the sources first**

First estimate the sources and then extract the conditional embeddings



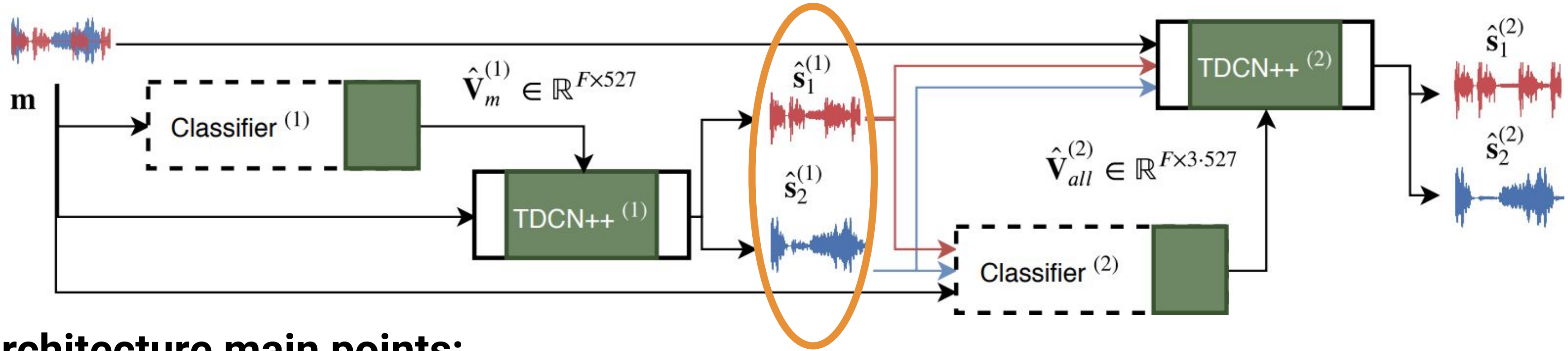
The Premise:

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

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**

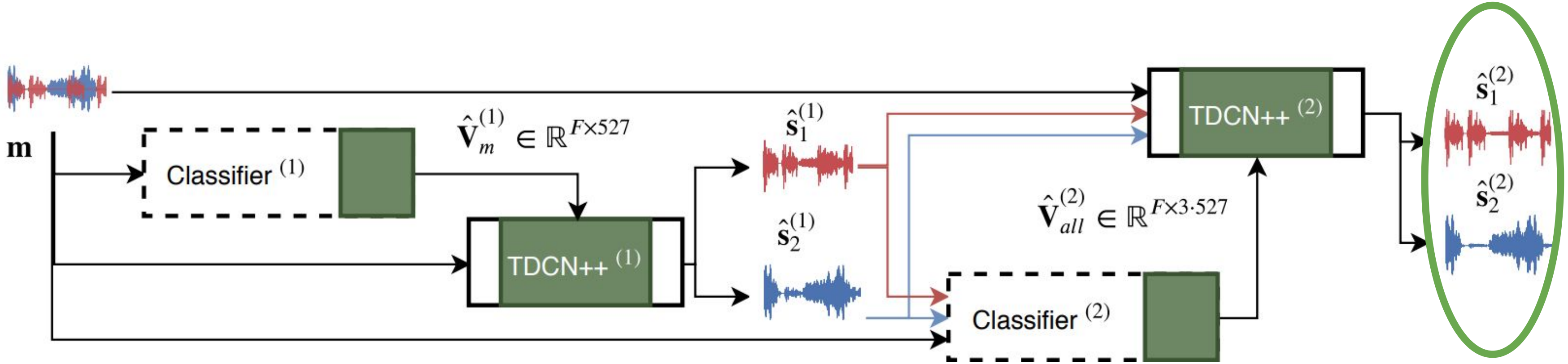
Iterative separation and refinement of embeddings (iTDCN++)



Architecture main points:

1. **Estimate the separated sources** and then extract the embeddings for both the estimates and the input mixture

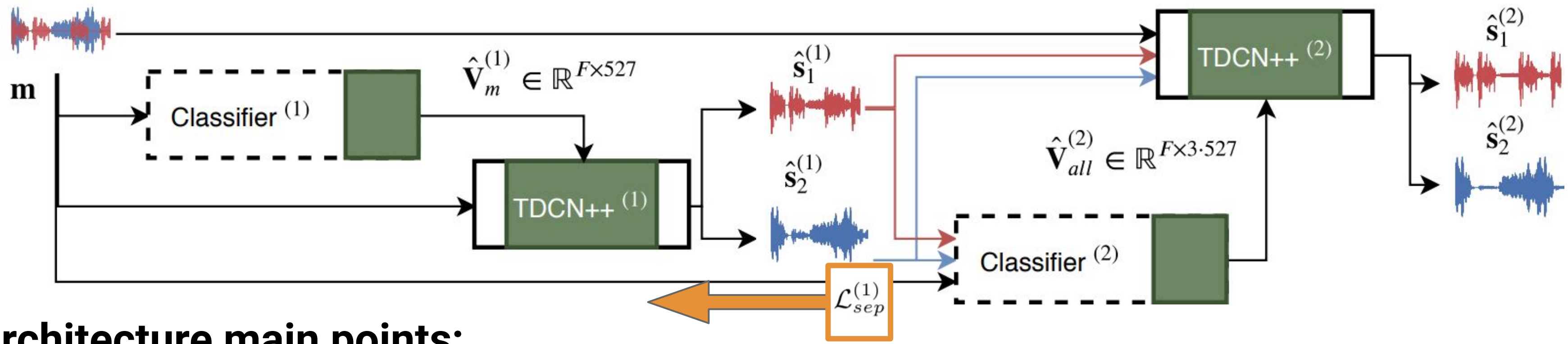
Iterative separation and refinement of embeddings (iTDCN++)



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**

Iterative separation and refinement of embeddings (iTDCN++)



Architecture main points:

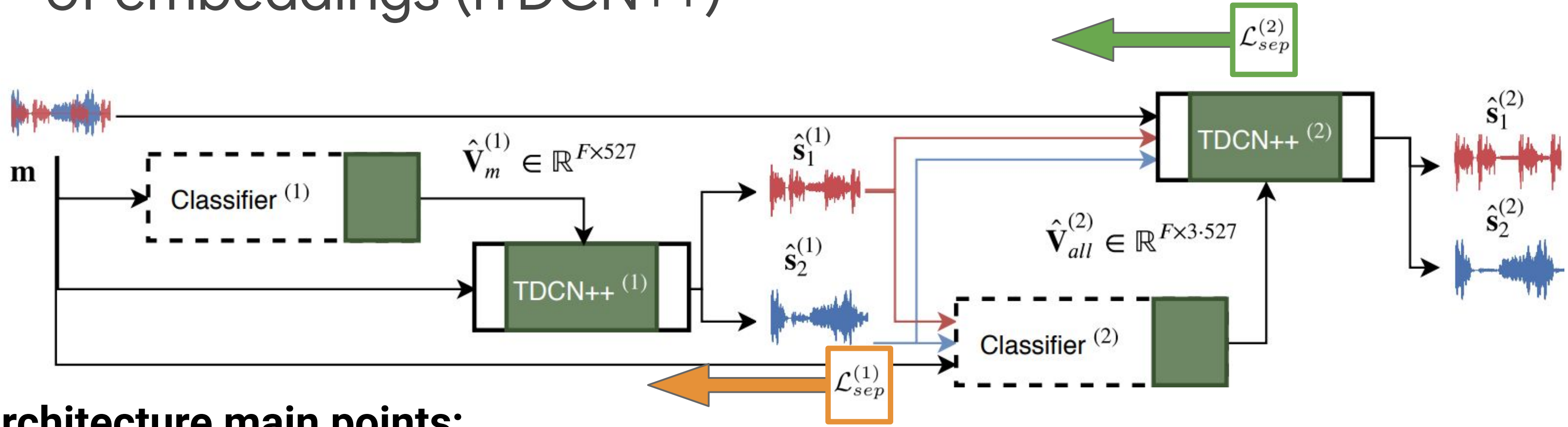
1. **Estimate the separated sources** and then extract the embeddings for both the estimates and the input mixture
2. Use the estimates and the embeddings for **making better the final separation**

Source separation losses:

1. First separation estimation:

$$\boxed{\mathcal{L}_{sep}^{(1)} = -SNR\left(\mathbf{s}_{p^*}, \hat{\mathbf{s}}^{(1)}\right)}$$

Iterative separation and refinement of embeddings (iTDCN++)



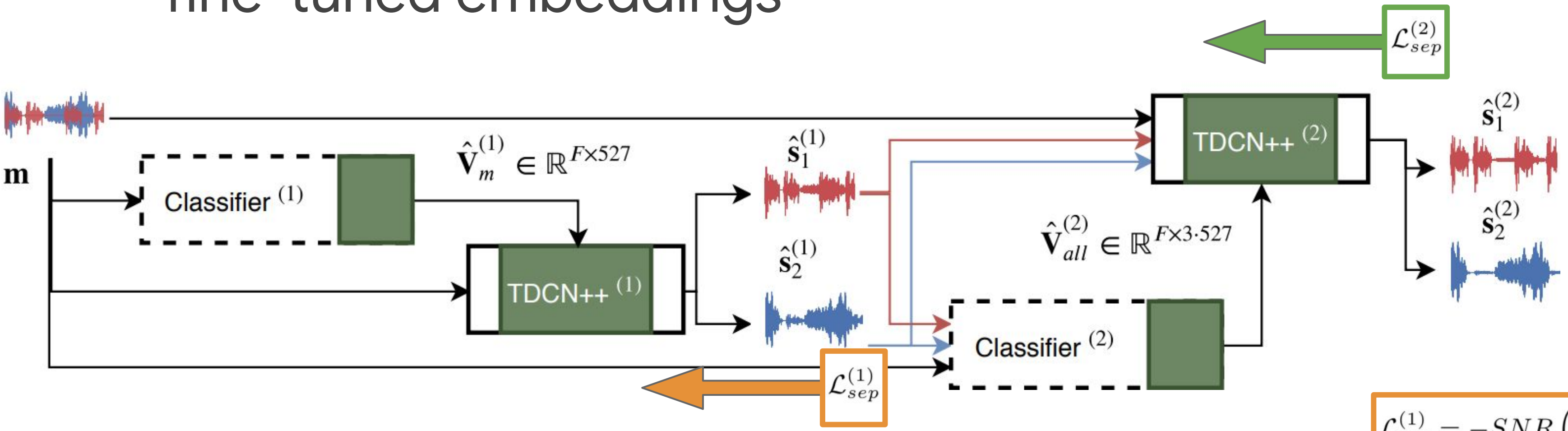
Architecture main points:

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**

Source separation losses:

1. First separation estimation: $\boxed{\mathcal{L}_{sep}^{(1)} = -SNR\left(\mathbf{s}_{p^*}, \hat{\mathbf{s}}^{(1)}\right)}$
2. Final separation estimation: $\boxed{\mathcal{L}_{sep}^{(2)} = -SNR\left(\mathbf{s}_{p^*}, \hat{\mathbf{s}}^{(2)}\right)}$

Guided Iterative separation and fine-tuned embeddings



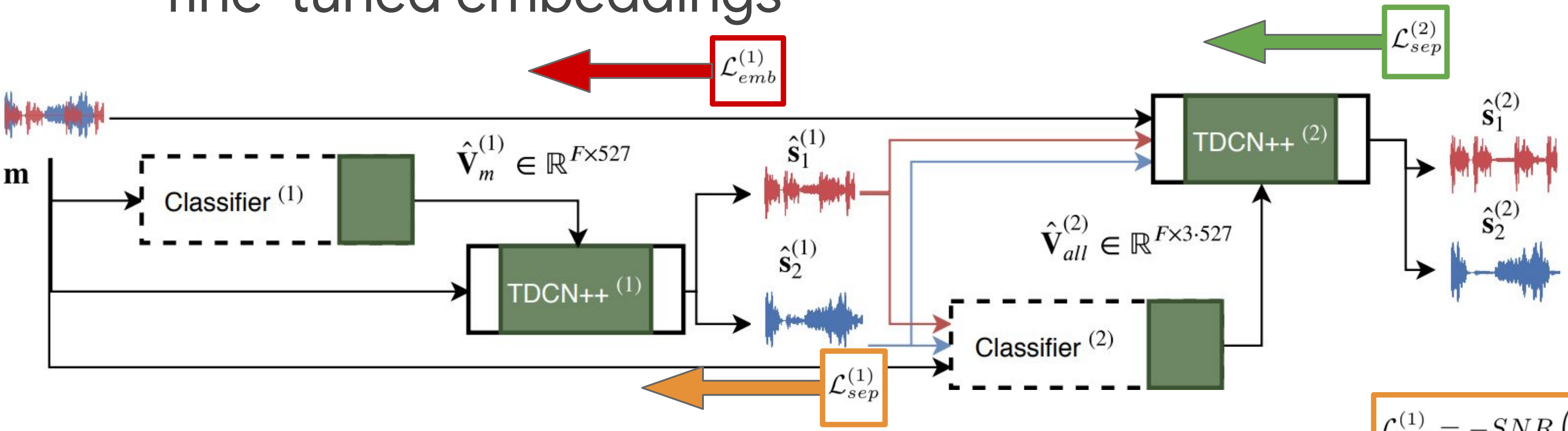
$$\mathcal{L}_{sep}^{(1)} = -SNR\left(\mathbf{s}_{p^*}, \hat{\mathbf{s}}^{(1)}\right)$$

$$\mathcal{L}_{sep}^{(2)} = -SNR\left(\mathbf{s}_{p^*}, \hat{\mathbf{s}}^{(2)}\right)$$

Idea: Use the "ideal" embeddings as targets

Embeddings Losses: sigmoid cross-entropy (SCE)

Guided Iterative separation and fine-tuned embeddings



$$\mathcal{L}^{(1)}_{sep} = -SNR\left(\mathbf{s}_{p^*}, \hat{\mathbf{s}}^{(1)}\right)$$

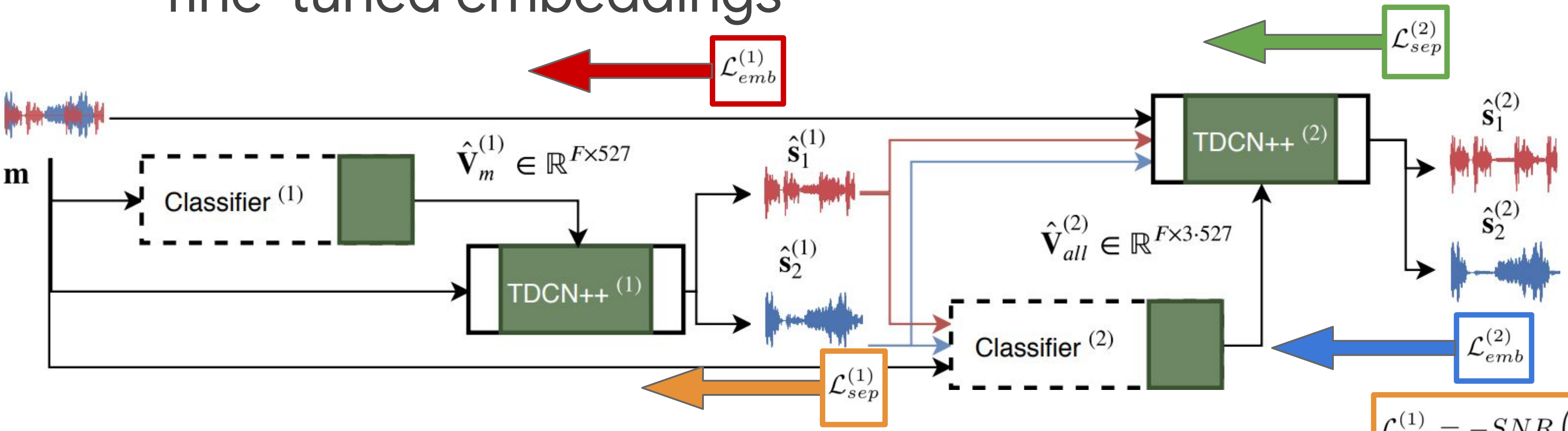
$$\mathcal{L}^{(2)}_{sep} = -SNR\left(\mathbf{s}_{p^*}, \hat{\mathbf{s}}^{(2)}\right)$$

Idea: Use the "ideal" embeddings as targets

Embeddings Losses: sigmoid cross-entropy (SCE)

- Making the **mixture embedding** look like the **soft OR embedding**: $\mathcal{L}^{(1)}_{emb} = SCE\left(\mathbf{V}^{p^*}_{or}, \hat{\mathbf{V}}^{(1)}_m\right)$

Guided Iterative separation and fine-tuned embeddings



$$\mathcal{L}_{sep}^{(1)} = -SNR\left(\mathbf{s}_{p^*}, \hat{\mathbf{s}}^{(1)}\right)$$

$$\mathcal{L}_{sep}^{(2)} = -SNR\left(\mathbf{s}_{p^*}, \hat{\mathbf{s}}^{(2)}\right)$$

Idea: Use the “ideal” embeddings as targets

Embeddings Losses: sigmoid cross-entropy (SCE)

- Making the **mixture embedding** look like the **soft OR embedding**: $\mathcal{L}_{emb}^{(1)} = SCE\left(\mathbf{V}_{or}^{p^*}, \hat{\mathbf{V}}_m^{(1)}\right)$
- Making the **sources embeddings** look like the **target ones**: $\mathcal{L}_{emb}^{(2)} = SCE\left(\mathbf{V}_{or}^{p^*}, \hat{\mathbf{V}}_m^{(2)}\right) + SCE\left(\mathbf{V}_{s}^{p^*}, \hat{\mathbf{V}}_s^{(2)}\right)$

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)

	Method	Embeddings Type	Embeddings Fine-tuning	STFT Val.	STFT Test	Learned Val.	Learned Test
Baselines	TDCN++ with no embeddings [8]	-	-	9.9	9.1	9.1	8.5
	iTDCN++ with no embeddings [8]	-	-	10.6	9.8	9.3	8.7
Proposed	Pretrained embeddings & TDCN++	mixture	-	10.3	9.4	9.4	8.6
	Fine-tuned embeddings & TDCN++	mixture	✓	10.2	9.4	9.3	8.5
	Guided fine-tuned embeddings & TDCN++	mixture	✓	10.3	9.4	9.4	8.6
	Pretrained embeddings & iTDCN++	all	-	10.8	9.9	9.9	9.0
	Fine-tuned embeddings & iTDCN++	all	✓	**11.1**	10.1	**10.1**	**9.2**
	Guided fine-tuned embeddings & iTDCN++	all	✓	**11.1**	**10.2**	10.0	9.1
Oracles	Pretrained embeddings & TDCN++	all	-	11.3	10.6	11.0	10.2
		soft-OR	-	11.4	10.6	10.7	10.1
	STFT binary mask	-	-	16.8	16.2	-	-

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)

	Method	Embeddings Type	Embeddings Fine-tuning	STFT Val.	STFT Test	Learned Val.	Learned Test
Baselines	TDCN++ with no embeddings [8]	-	-	9.9	$\boxed{9.1}$	9.1	$\boxed{8.5}$
	iTDCN++ with no embeddings [8]	-	-	10.6	9.8	9.3	8.7
Proposed	Pretrained embeddings & TDCN++	mixture	-	10.3	9.4	9.4	8.6
	Fine-tuned embeddings & TDCN++	mixture	✓	10.2	$\boxed{9.4}$	9.3	$\boxed{8.5}$
	Guided fine-tuned embeddings & TDCN++	mixture	✓	10.3	9.4	9.4	8.6
	Pretrained embeddings & iTDCN++	all	-	10.8	9.9	9.9	9.0
	Fine-tuned embeddings & iTDCN++	all	✓	**11.1**	10.1	**10.1**	**9.2**
	Guided fine-tuned embeddings & iTDCN++	all	✓	**11.1**	**10.2**	10.0	9.1
Oracles	Pretrained embeddings & TDCN++	all	-	11.3	10.6	11.0	10.2
		soft-OR	-	11.4	10.6	10.7	10.1
	STFT binary mask	-	-	16.8	16.2	-	-

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)

	Method	Embeddings		STFT		Learned	
		Type	Fine-tuning	Val.	Test	Val.	Test
Baselines	TDCN++ with no embeddings [8]	-	-	9.9	9.1	9.1	8.5
	iTDCN++ with no embeddings [8]	-	-	10.6	$\boxed{9.8}$	9.3	$\boxed{8.7}$
Proposed	Pretrained embeddings & TDCN++	mixture	-	10.3	9.4	9.4	8.6
	Fine-tuned embeddings & TDCN++	mixture	✓	10.2	9.4	9.3	8.5
	Guided fine-tuned embeddings & TDCN++	mixture	✓	10.3	9.4	9.4	8.6
	Pretrained embeddings & iTDCN++	all	-	10.8	9.9	9.9	9.0
	Fine-tuned embeddings & iTDCN++	all	✓	**11.1**	10.1	**10.1**	$\boxed{\textbf{9.2}}$
	Guided fine-tuned embeddings & iTDCN++	all	✓	**11.1**	$\boxed{\textbf{10.2}}$	10.0	9.1
Oracles	Pretrained embeddings & TDCN++	all	-	11.3	10.6	11.0	10.2
		soft-OR	-	11.4	10.6	10.7	10.1
	STFT binary mask	-	-	16.8	16.2	-	-

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)

	Method	Embeddings Type	Embeddings Fine-tuning	STFT Val.	STFT Test	Learned Val.	Learned Test
Baselines	TDCN++ with no embeddings [8]	-	-	9.9	9.1	9.1	8.5
	iTDCN++ with no embeddings [8]	-	-	10.6	9.8	9.3	8.7
Proposed	Pretrained embeddings & TDCN++	mixture	-	10.3	9.4	9.4	8.6
	Fine-tuned embeddings & TDCN++	mixture	✓	10.2	9.4	9.3	8.5
	Guided fine-tuned embeddings & TDCN++	mixture	✓	10.3	9.4	9.4	8.6
	Pretrained embeddings & iTDCN++	all	-	10.8	9.9	9.9	9.0
	Fine-tuned embeddings & iTDCN++	all	✓	**11.1**	10.1	**10.1**	**9.2**
	Guided fine-tuned embeddings & iTDCN++	all	✓	**11.1**	**10.2**	10.0	9.1
Oracles	Pretrained embeddings & TDCN++	all	-	11.3	10.6	11.0	10.2
		soft-OR	-	11.4	10.6	10.7	10.1
	STFT binary mask	-	-	16.8	16.2	-	-

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

Google AI