# Listening to Multi-talker Conversations

## Modular and End-to-end Perspectives

**Desh Raj**
**Graduate Board Oral Examination**
**May 4, 2022**

# Motivation

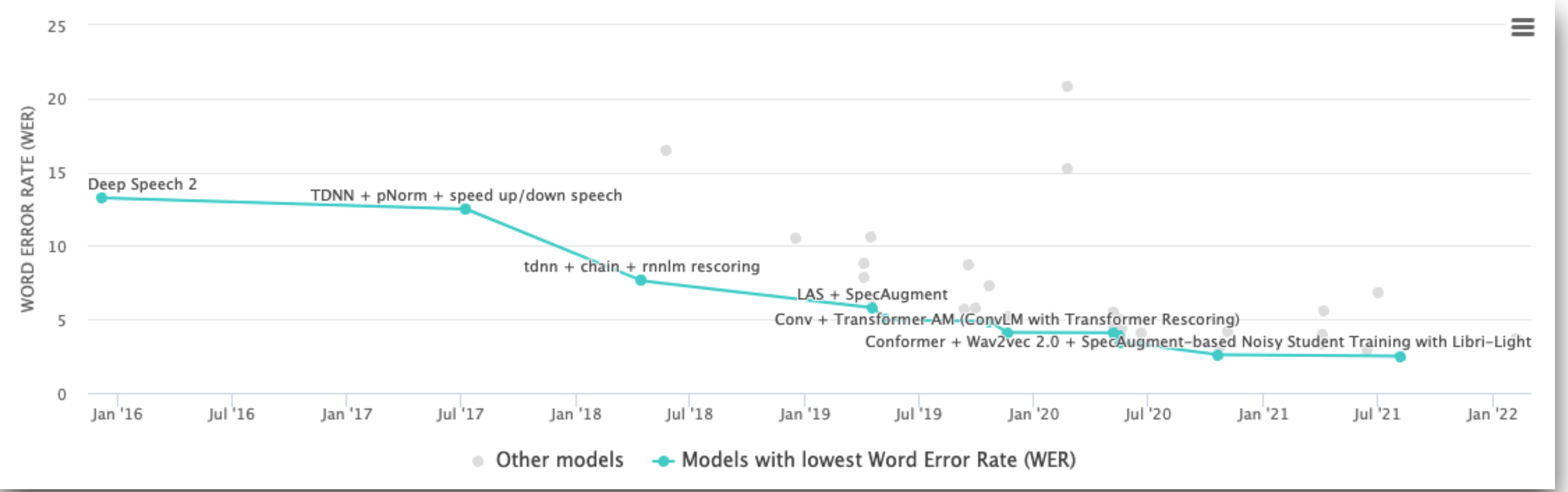
OCTOBER 20, 2020

## AI outperforms humans in speech recognition

by Monika Landgraf, Karlsruhe Institute of Technology

## Microsoft claims new speech recognition record, achieving a super-human 5.1% error rate

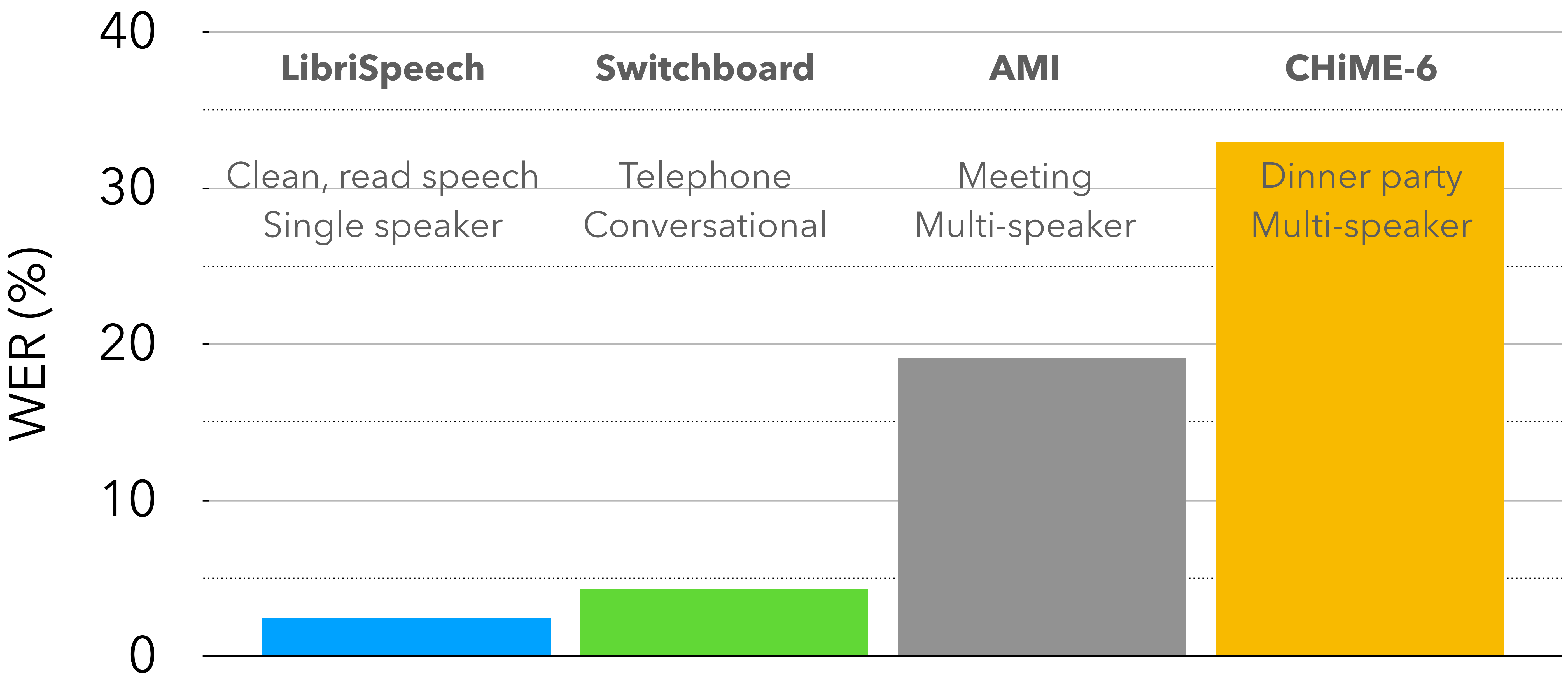
BY TODD BISHOP on August 20, 2017 at 7:44 pm

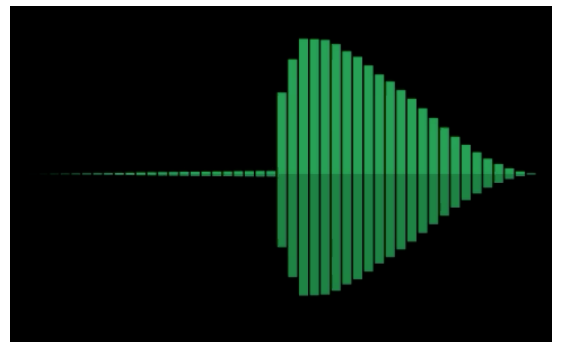



https://paperswithcode.com/sota/speech-recognition-on-librispeech-test-other

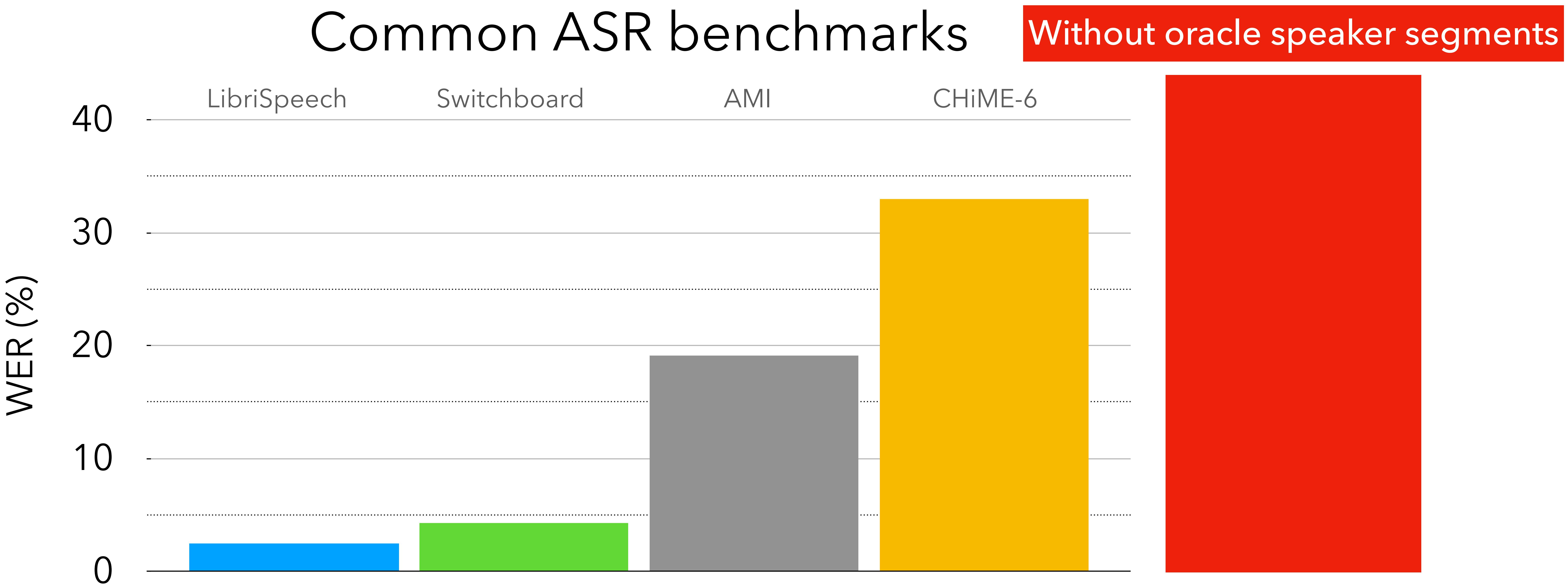
# Motivation

Common ASR benchmarks



# Motivation




**What changed?**

- Conversational speech
- Far-field audio: noise and reverberation
- Overlapping speakers

# Motivation

**Single-user applications**

Smart Assistants

Language Learning

Customer Service

Voice-based Search

**Multi-user applications**

Meeting summaries

Collaborative Learning

Cocktail-party Problem

# Problem Statement

## Multi-talker speaker-attributed ASR

- **Input:** long unsegmented (possibly multi-channel) recording containing multiple speakers.
- **Output:**
  - Transcription of the recording (speech recognition)
  - Speaker attribution (diarization)
  - Additional constraints: streaming, i.e., real-time transcription
- We specifically look at “meetings”: LibriCSS, AMI, AliMeeting

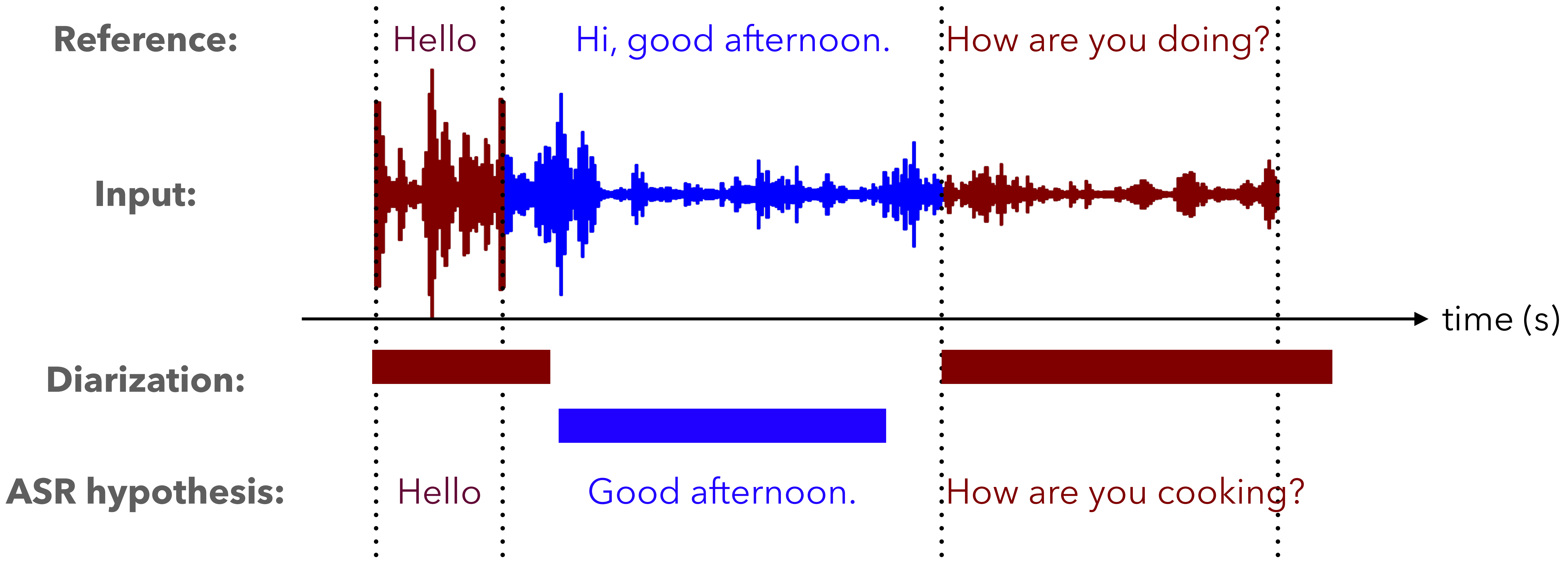
# Problem Statement

## Corpora

| **Corpus Name** | **LibriCSS** [1] | **AMI** [2] | **AliMeeting** [3] |
|---|---|---|---|
| **Session length** | 10 minutes | 30-45 minutes | 15-30 minutes |
| **Total size of corpus** | 10 hours | 100 hours | 120 hours |
| **Microphones available** | 7-channel circular array | 2 linear arrays with 8 channels each + headset | 8-channel circular array + headset mics |
| **Number of speakers** | 8 | 4 | 2-4 |
| **Overlap ratio** | 0 to 40% | ~20% | ~35% |
| **Language** | English | English | Mandarin |
| | Simulated (replayed) | Real meetings | Real meetings |

# Problem Statement

## Evaluation Metrics




**Diarization Error Rate (DER)**

$$\frac{\text{Missed speech + False alarms + Speaker confusion}}{\text{Total speaking time}}$$

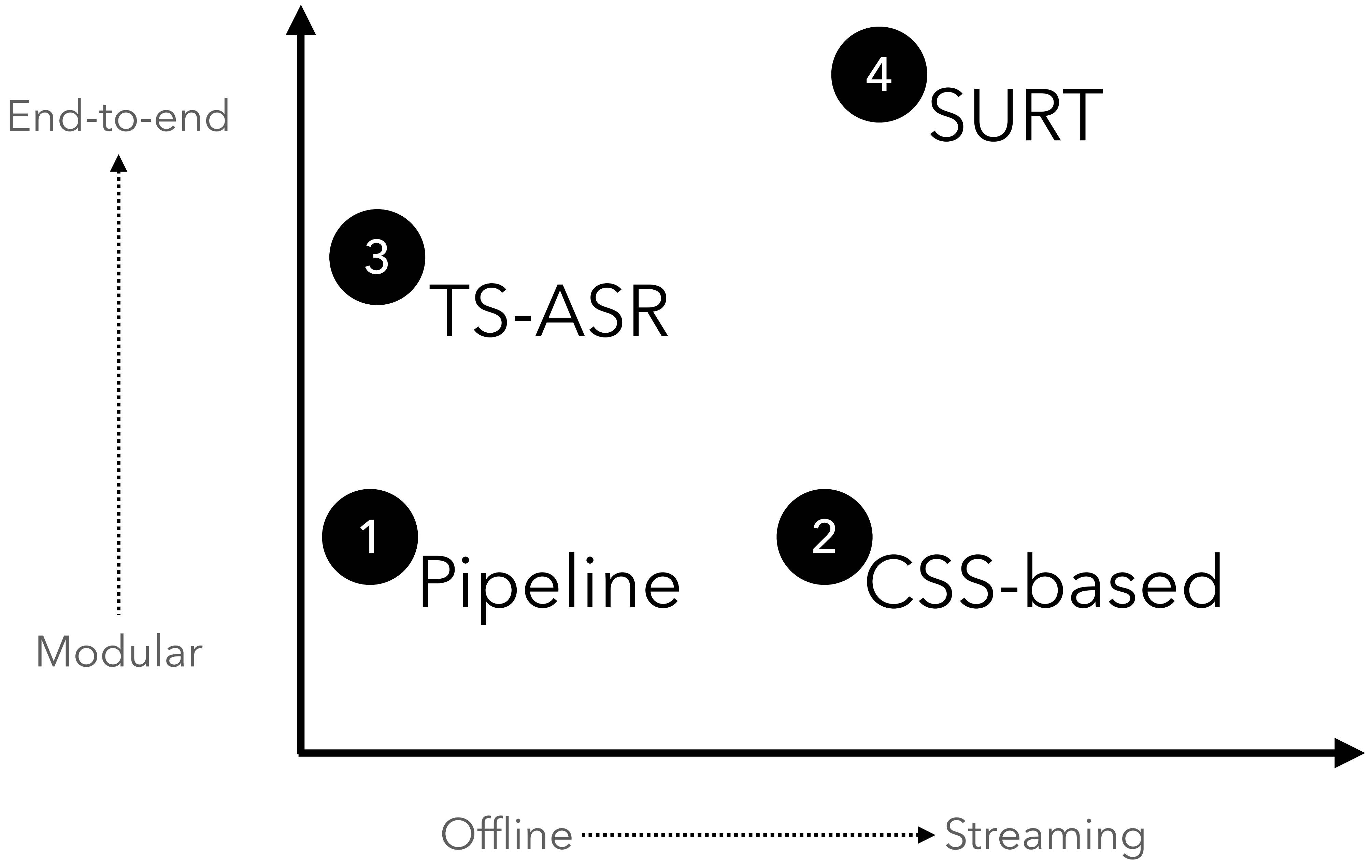
**Concatenated minimum permutation Word Error Rate (cpWER)**

| | | |
|---|---|---|
| **Concatenated reference:** | Hello How are you doing? | Hi, good afternoon. |
| **Concatenated hypothesis:** | Hello How are you cooking? | Good afternoon. |

Compute average WER for all permutations of speakers and return minimum

# How to solve this problem?

## Modular and end-to-end approaches

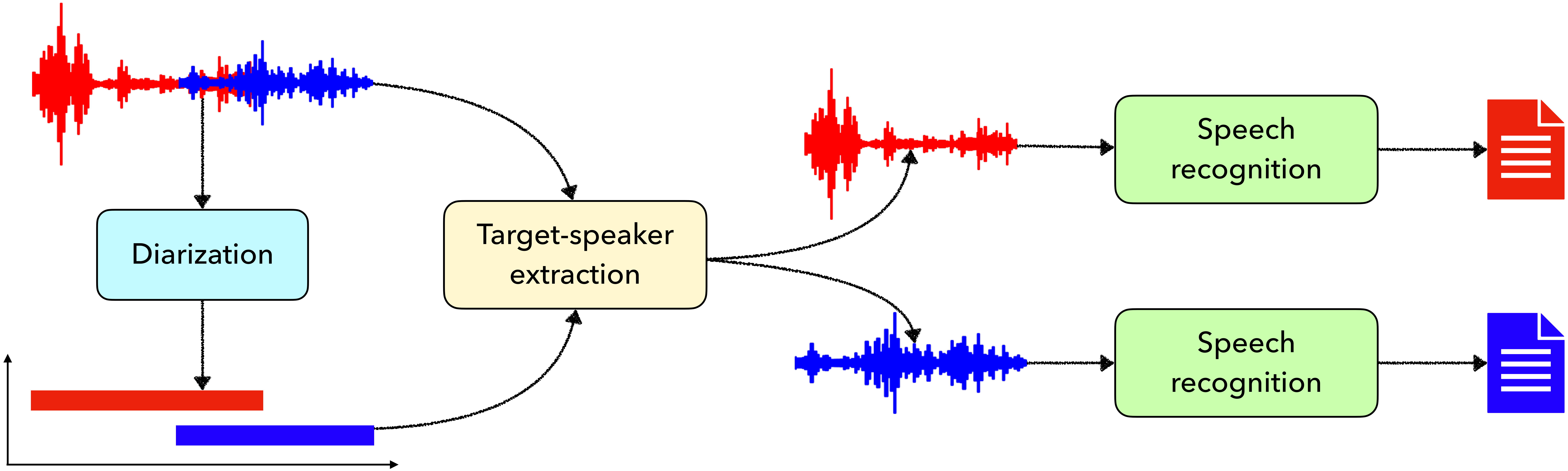



**KEY COMPONENTS**

| | |
|---|---|
| Speaker Diarization | Who spoke when? |
| Speech Recognition | Transcribe audio |
| Speech Separation | Separate overlapping speech |

# Modular Perspective

## Pipeline approach: the CHiME-6 challenge [4]




- Need to assign overlapping speech to speakers

- Multi-channel **guided source separation** (GSS)
- Unsupervised target-speaker extraction method
- Works well if segments are accurate

- Can leverage advances in single-speaker ASR methods
- Mismatch between train and test?
- Inaccurate segment boundaries can cause insertion/ deletion errors

# Modular Perspective

## Overlap-aware diarization [5]

- Conventional diarization methods make single-speaker assumption: bad for both GSS and ASR modules
- Novel method for overlap assignment with spectral clustering
- Results on **LibriCSS**:

| Method | DER | cpWER |
|---|---|---|
| **Spectral clustering** | 14.9 | 17.4 |
| **+ overlap assignment** | **11.3** | **14.3** |

# Modular Perspective

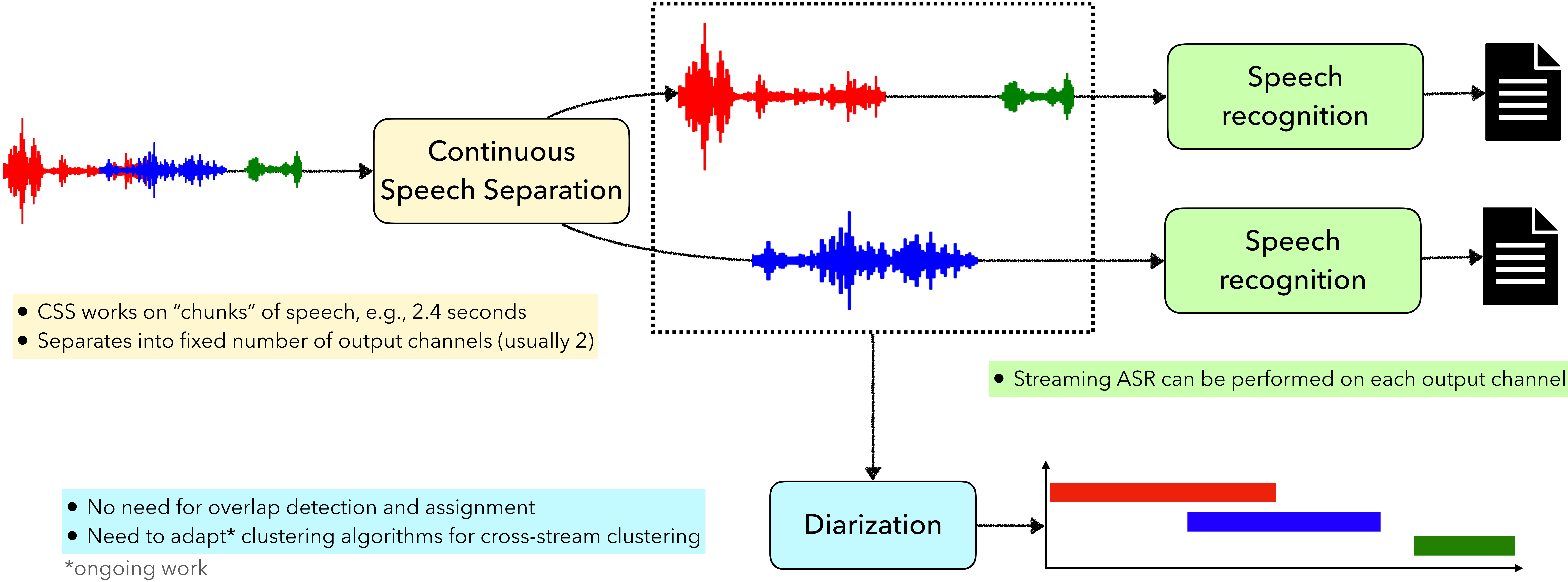
## Simultaneous systems based on CSS [6]

- Pipeline system is *offline*
- Needs *special methods* for overlap-aware diarization

💡 Use **speech separation** front-end

# Modular Perspective

## Simultaneous systems based on CSS [6]




- CSS works on “chunks” of speech, e.g., 2.4 seconds
- Separates into fixed number of output channels (usually 2)

- Streaming ASR can be performed on each output channel

- No need for overlap detection and assignment
- Need to adapt* clustering algorithms for cross-stream clustering

*ongoing work

# Modular Perspective

## Simultaneous systems based on CSS [6]

- How does it compare with the pipeline system?
- Performance on **LibriCSS**:

| Method | DER | cpWER |
|---|---|---|
| **Pipeline system** | **11.3** | 14.3 |
| **CSS-based system** | 14.1 | **12.7** |

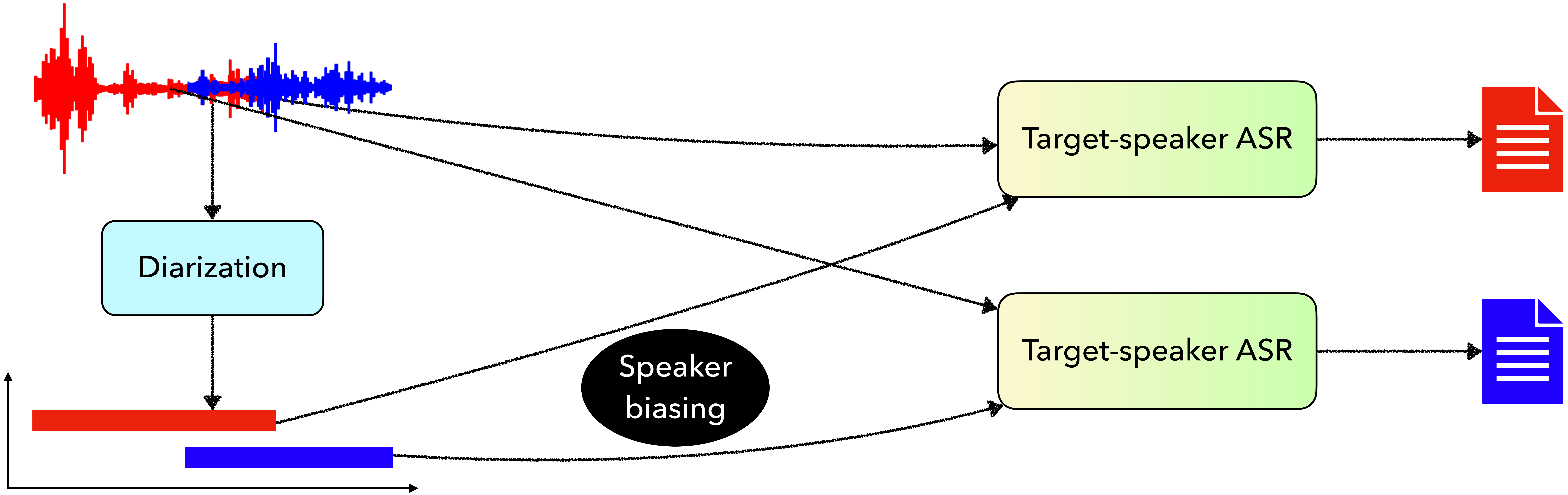
# End-to-end Perspective

## Separation-free approach with target-speaker ASR

- It is hard to train separation networks for partially overlapped recordings.
- Adds overhead since we do not need to produce separated audio
- Can we build “separation-free” systems?

# End-to-end Perspective

## Separation-free approach with target-speaker ASR




- Overlap-aware diarization, similar to "pipeline" system
- Extract speaker embedding and use for biasing the TS-ASR module

- Combines target-speaker extraction and ASR components
- Previous methods: SpeakerBeam, VoiceFilter
- Use transducer-based TS-ASR*

*proposed

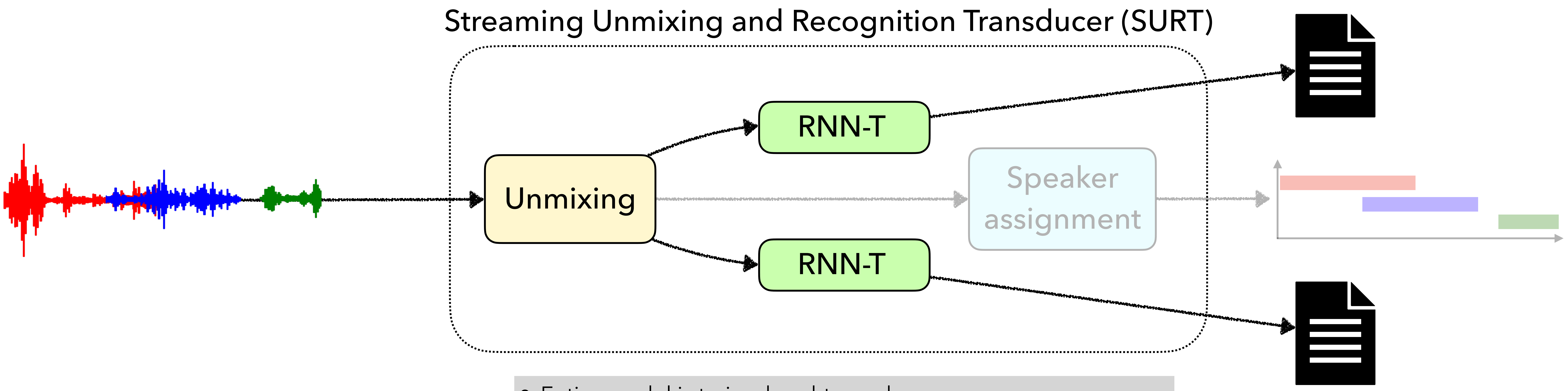
# End-to-end Perspective

## Separation-free approach with target-speaker ASR

- The TS-ASR based system is also *offline* since it depends on the diarization output
- How to build a *fully end-to-end* system for multi-talker ASR?

# End-to-end Perspective

## Continuous streaming multi-talker ASR with SURT [7]




- Entire model is trained end-to-end
- Current model supports streaming multi-talker ASR without speaker assignment
- Speaker assignment: transcribe-to-diarize

# Exercise: Fill in the Blanks

## Benchmarking the systems on public corpora

✓ Exists in literature

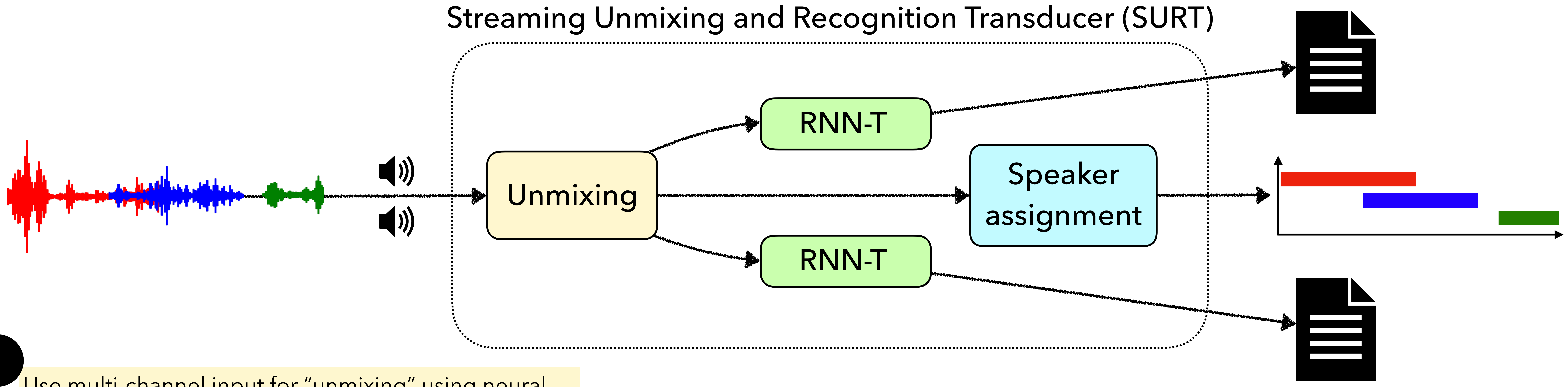
✗ No previous studies

⚙ Finished work

| | System | LibriCSS | AMI | AliMeeting |
|---|---|---|---|---|
| Originally for CHiME-6 | **Pipeline** | ⚙ | ⚙ | ✗ |
| Lot of work with LibriCSS | **CSS-based** | ✓ | ✗ | ✗ |
| Previous work uses WSJ-Mix | **TS-ASR** | ✗ | ✗ | ✗ |
| | **SURT** | ✓ | ✗ | ✗ |

# Advances in SURT

## Multi-channel models, graph-PIT, and self-supervised learning




1 Use multi-channel input for “unmixing” using neural MVDR beamformer [8]

2 Use self-supervised encoders like Wav2Vec 2.0 [10]

3 Use more flexible training objectives like Graph-PIT instead of HEAT [9]

# Review

## What we hope to achieve at the end of this thesis

- **Formalize** the multi-talker ASR task and review popular approaches from literature
- **Benchmark** the systems on public datasets and analyze pros and cons
- Propose **new strategies** for challenges within these systems (overlap-aware diarization, train-test mismatch for ASR, etc.)
- Develop **transducer-based** end-to-end multi-talker ASR models for continuous and streaming recognition

# References


1. Chen, Zhuo et al. "Continuous Speech Separation: Dataset and Analysis." IEEE ICASSP 2020.
2. Carletta, Jean et al. "The AMI Meeting Corpus: A Pre-announcement." MLMI (2005).
3. Yu, Fan et al. "M2MeT: The ICASSP 2022 Multi-Channel Multi-Party Meeting Transcription Challenge." ArXiv abs/2110.07393 (2021).
4. Arora, Ashish et al. "The JHU Multi-Microphone Multi-Speaker ASR System for the CHiME-6 Challenge." ArXiv abs/2006.07898 (2020).
5. **Raj, Desh** et al. "Multi-Class Spectral Clustering with Overlaps for Speaker Diarization." IEEE SLT 2021.
6. **Raj, Desh** et al. "Integration of Speech Separation, Diarization, and Recognition for Multi-Speaker Meetings: System Description, Comparison, and Analysis." IEEE SLT 2021.
7. **Raj, Desh** et al. "Continuous Streaming Multi-Talker ASR with Dual-path Transducers." 2022 IEEE ICASSP.
8. Zhang, Zhuo-huang et al. "All-neural beamformer for continuous speech separation." ArXiv abs/2110.06428 (2021).
9. von Neumann, Thilo et al. "Graph-PIT: Generalized permutation invariant training for continuous separation of arbitrary numbers of speakers." Interspeech (2021).
10. Baevski, Alexei et al. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations." ArXiv abs/2006.11477 (2020).

# Extra Slides

# Overlap-aware Spectral Clustering

# Speaker Diarization

## “Clustering-based” systems

- **Key idea:** formulate Diarization as a clustering problem
- Cluster small segments of audio
- Each cluster represents a distinct speaker

Basu, J., Khan, S., Roy, R., Pal, M., Basu, T., Bepari, M.S., & Basu, T.K. (2016). An overview of speaker diarization: Approaches, resources and challenges.

Tranter, S., & Reynolds, D. (2006). An overview of automatic speaker diarization systems. IEEE Transactions on Audio, Speech, and Language Processing.

# Clustering-based diarization

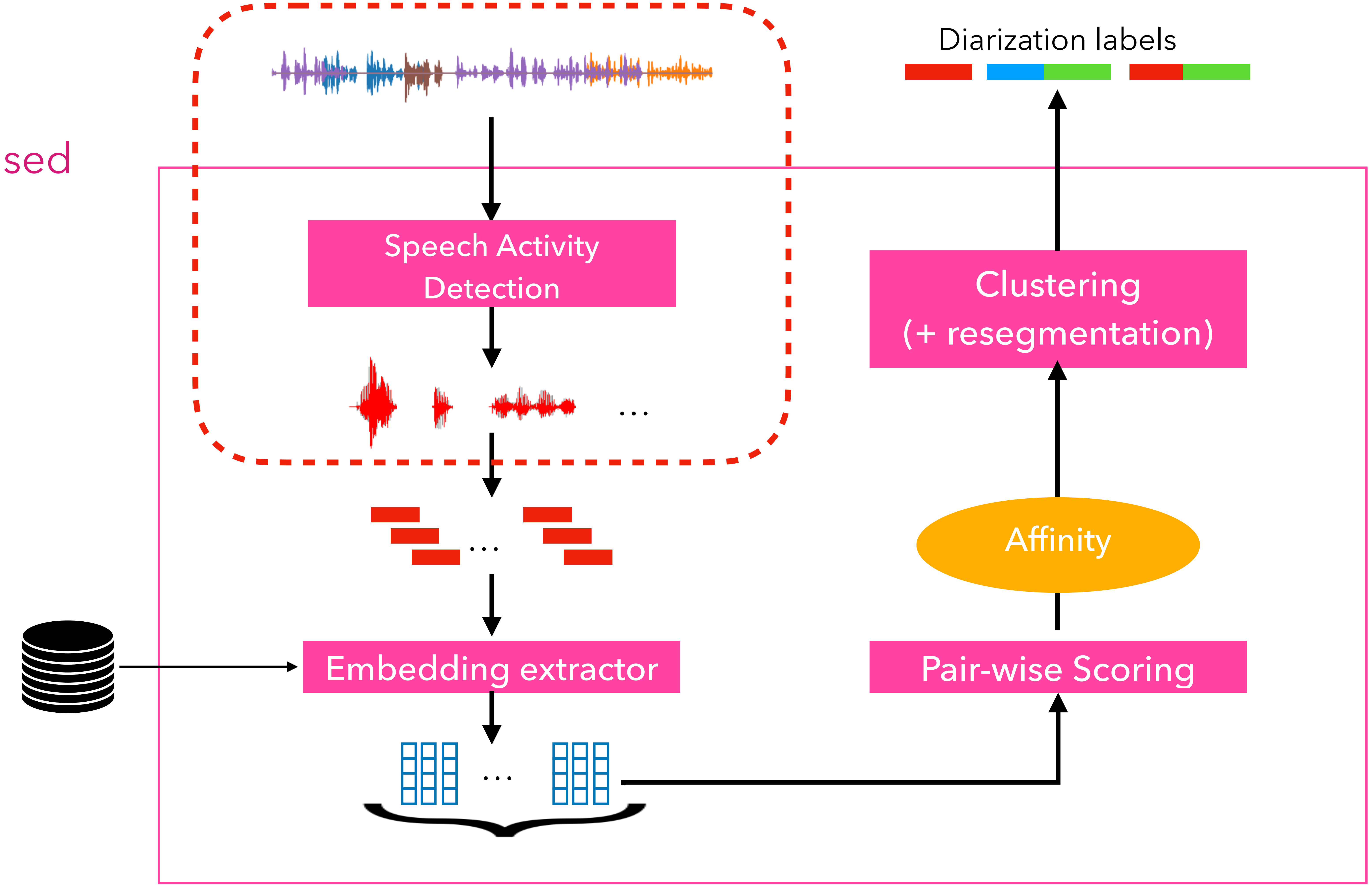
## SAD extracts speech segments from recordings

Spectral energy-based

GMM-based

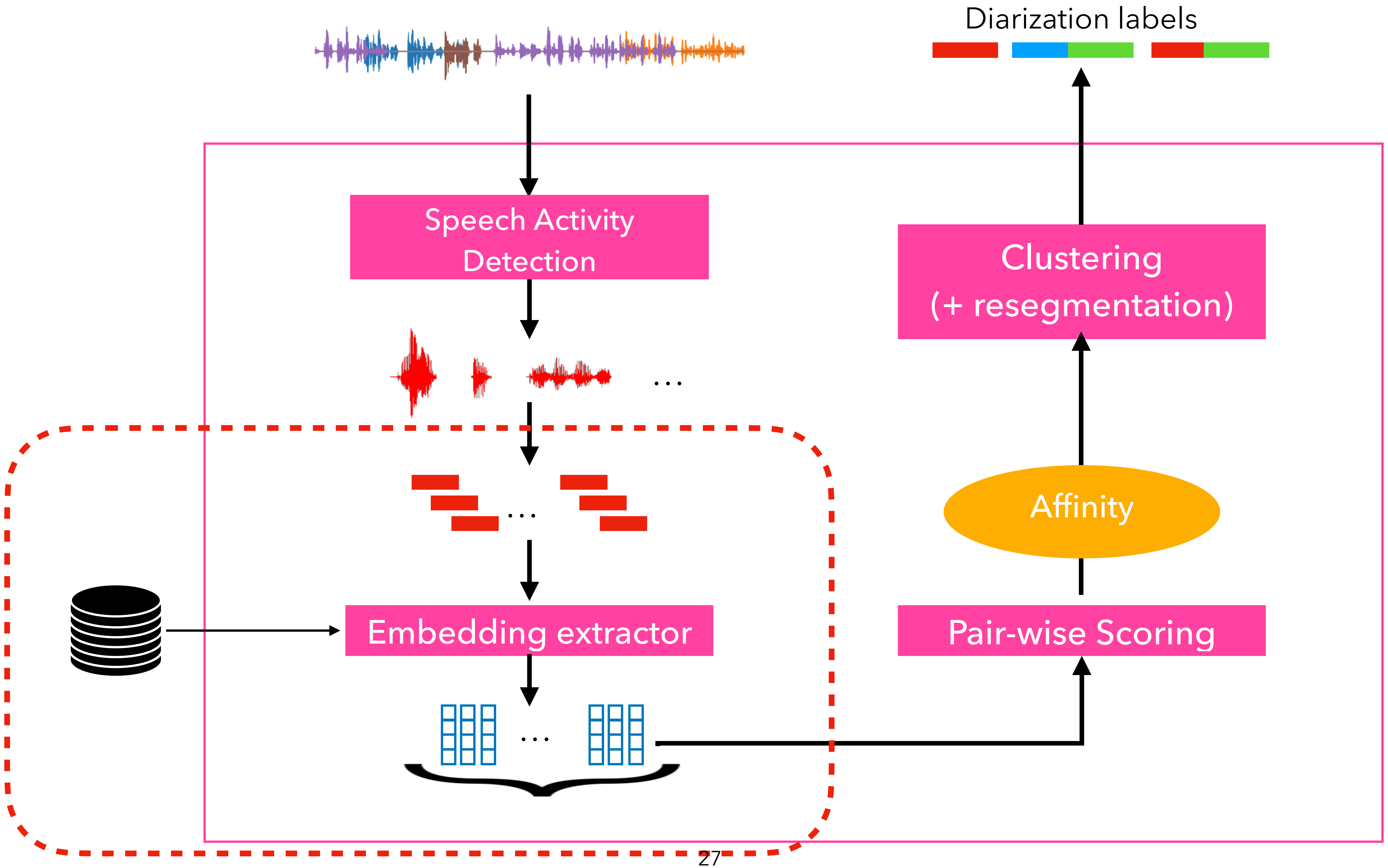
Hybrid HMM-DNN

End-to-end SAD



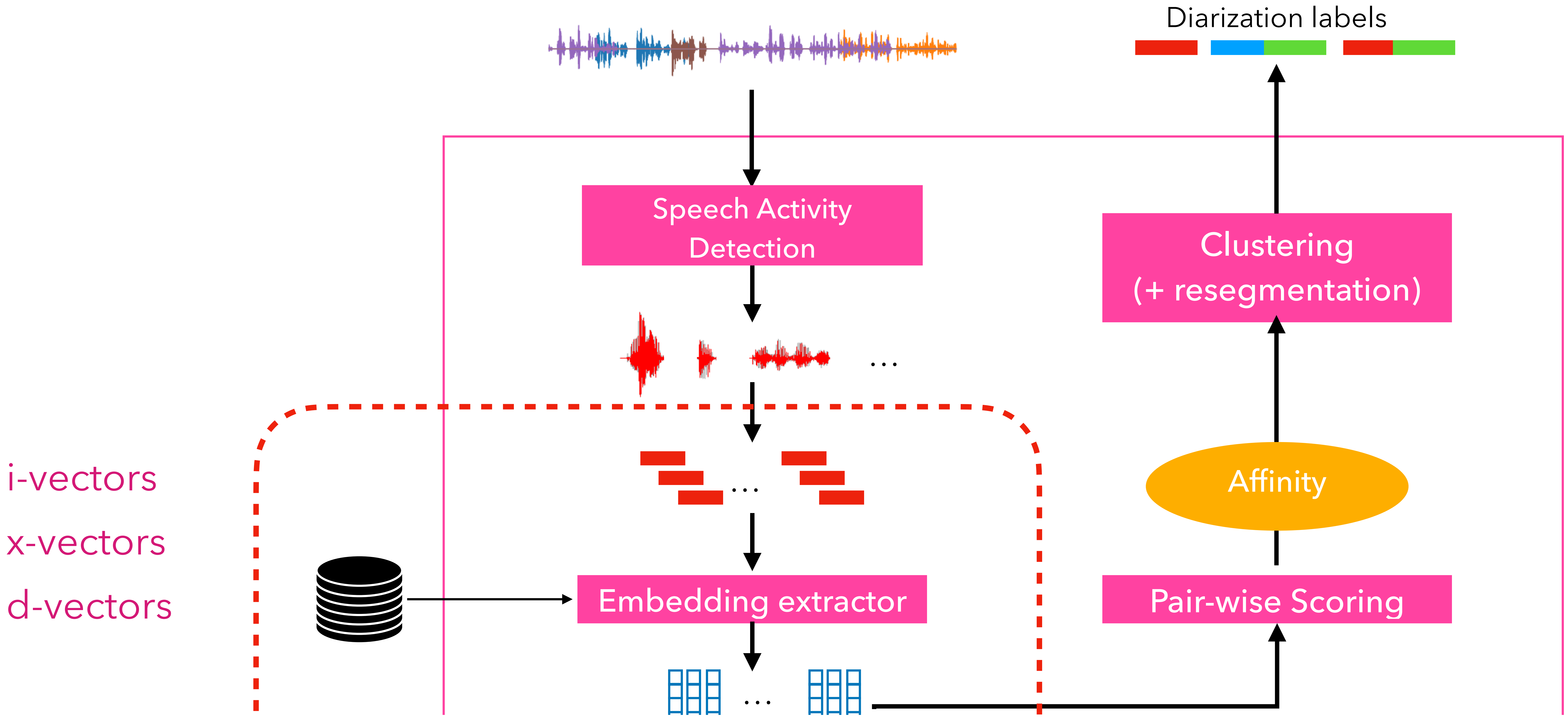
# Clustering-based diarization

## Embeddings extracted for small subsegments



# Clustering-based diarization

## Embeddings extracted for small subsegments

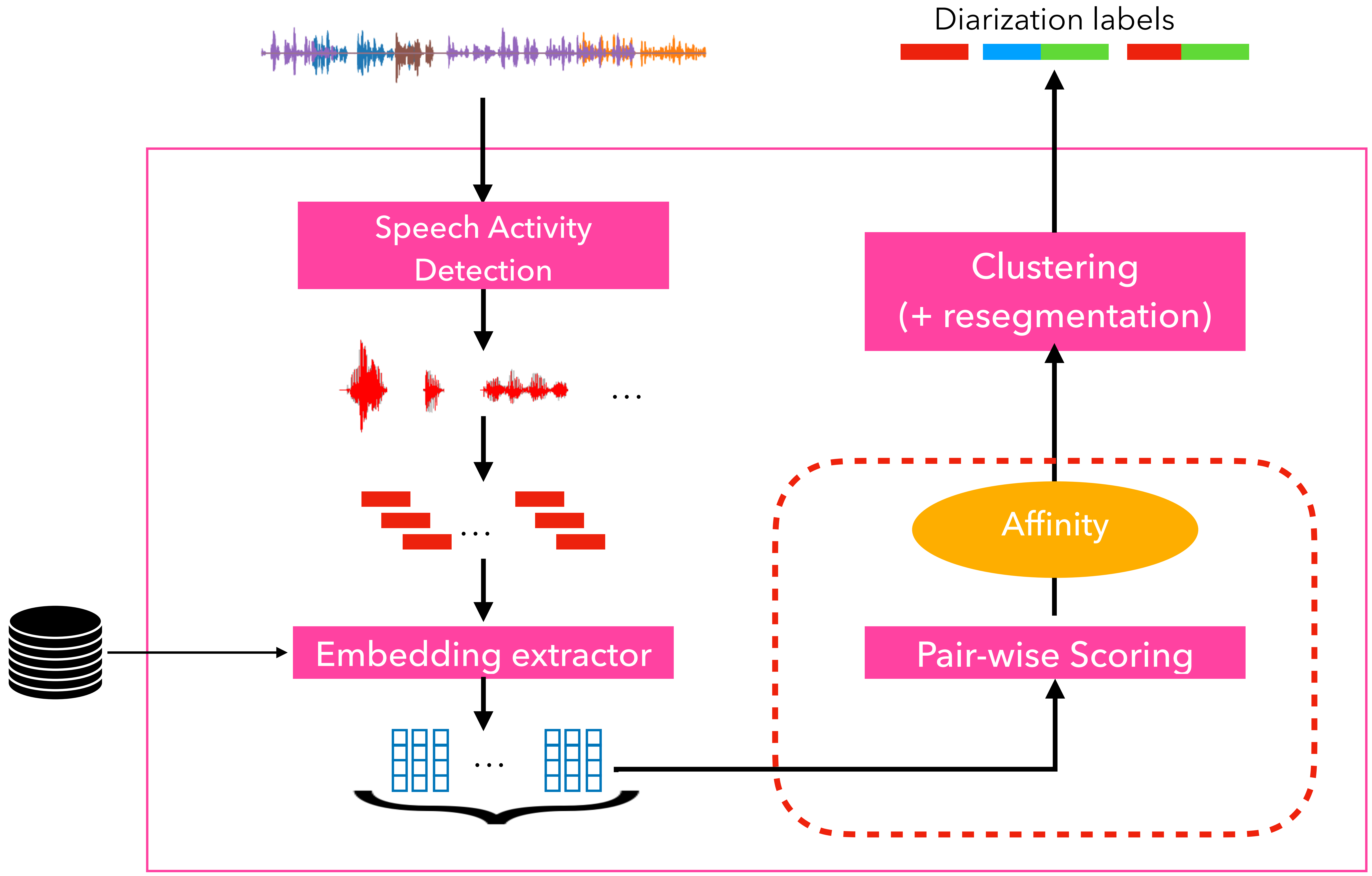




Dehak, N., et al (2011). Front-End Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing*.
Snyder, D., et al. (2018). X-Vectors: Robust DNN Embeddings for Speaker Recognition. *2018 IEEE ICASSP*.
Variani, E.,et al. (2014). Deep neural networks for small footprint text-dependent speaker verification. 2014 IEEE ICASSP.

# Clustering-based diarization

## Pair-wise scoring of subsegments




PLDA scoring
Cosine scoring

Sell, G., & Garcia-Romero, D. (2014). Speaker diarization with PLDA i-vector scoring and unsupervised calibration. *2014 IEEE Spoken Language Technology Workshop (SLT).*

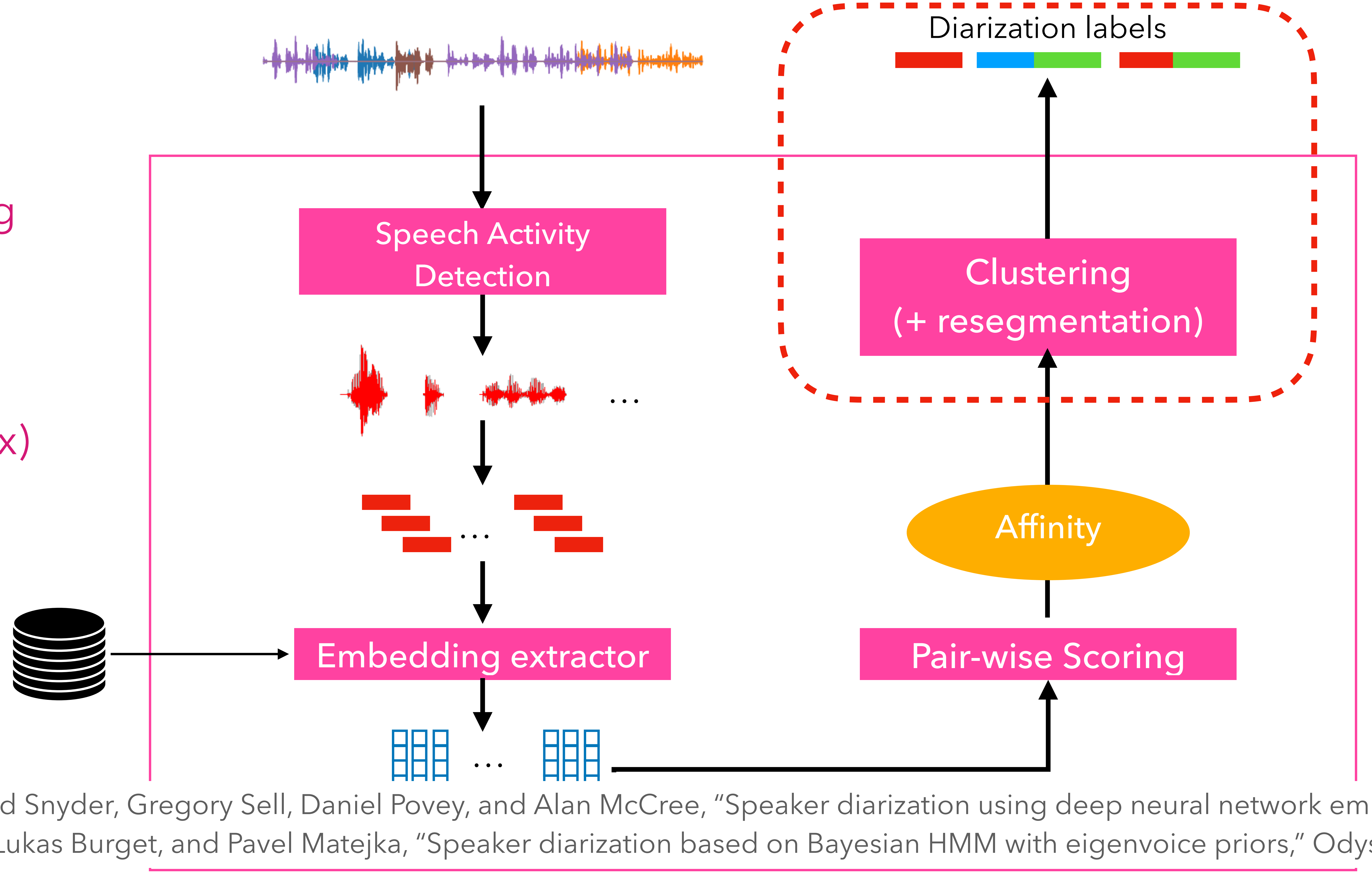
# Clustering-based diarization

## Clustering based on the affinity matrix, followed by optional resegmentation

Agglomerative hierarchical clustering

Spectral clustering

Variational Bayes (VBx)




Daniel Garcia-Romero, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree, "Speaker diarization using deep neural network embeddings," ICASSP 2017.
Mireia Dîez, Lukas Burget, and Pavel Matejka, "Speaker diarization based on Bayesian HMM with eigenvoice priors," Odyssey 2018.

# Clustering-based diarization

## How well does it perform?

- **Winning system in DIHARD I (2018) and II (2019)**
- DIHARD contains "hard" Diarization evaluation with recordings from several domains
- But Diarization error rates (DER) still high: 37% in DIHARD I and 27% in DIHARD II

$$\text{DER} = \frac{\textbf{Missed speech + False alarm + Speaker error}}{\textbf{Total speaking time}}$$

Sell, G., et al. (2018). Diarization is Hard: Some Experiences and Lessons Learned for the JHU Team in the Inaugural DIHARD Challenge. *INTERSPEECH 2018*.
Landini, F., et al. (2020). BUT System for the Second Dihard Speech Diarization Challenge. *IEEE ICASSP 2020*.

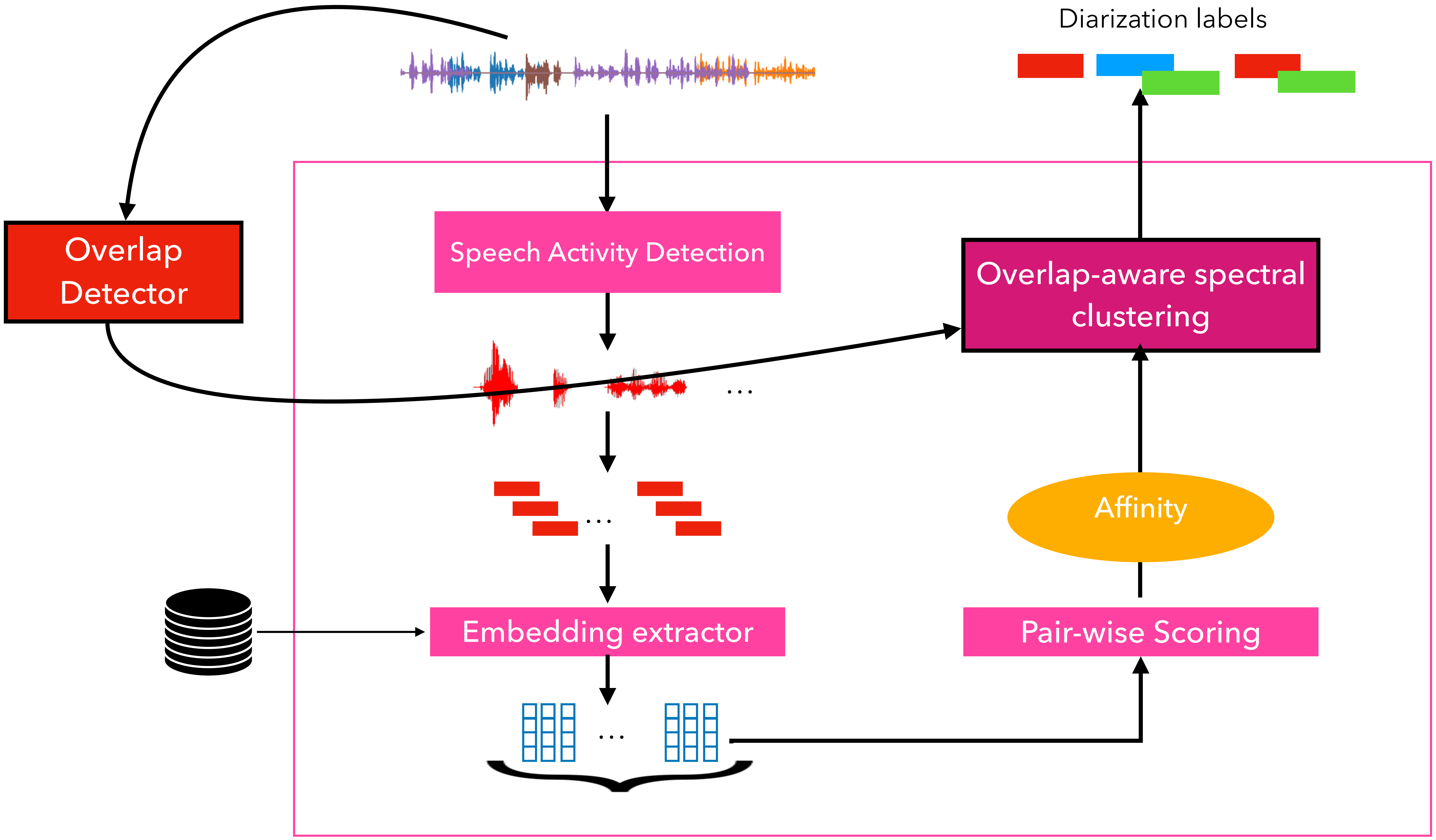
# Clustering paradigm assumes single-speaker segments

## So overlapping speakers are completely ignored!

*"Roughly **8% of the absolute error** in our systems was from overlapping speech … it will likely require a **complete rethinking of the diarization process** … This is an important direction, but could not be addressed …"*
***- JHU team (2018)***

*"Given the current performance of the systems, the **overlapped speech gains more relevance** … **more than 50% of the DER** in our best systems … has to be addressed in the future …"*
***- BUT team (2019)***

# Overlap-aware spectral clustering




Raj, D., Huang, Z., & Khudanpur, S. (2021). Multi-class Spectral Clustering with Overlaps for Speaker Diarization. *IEEE SLT 2021*.

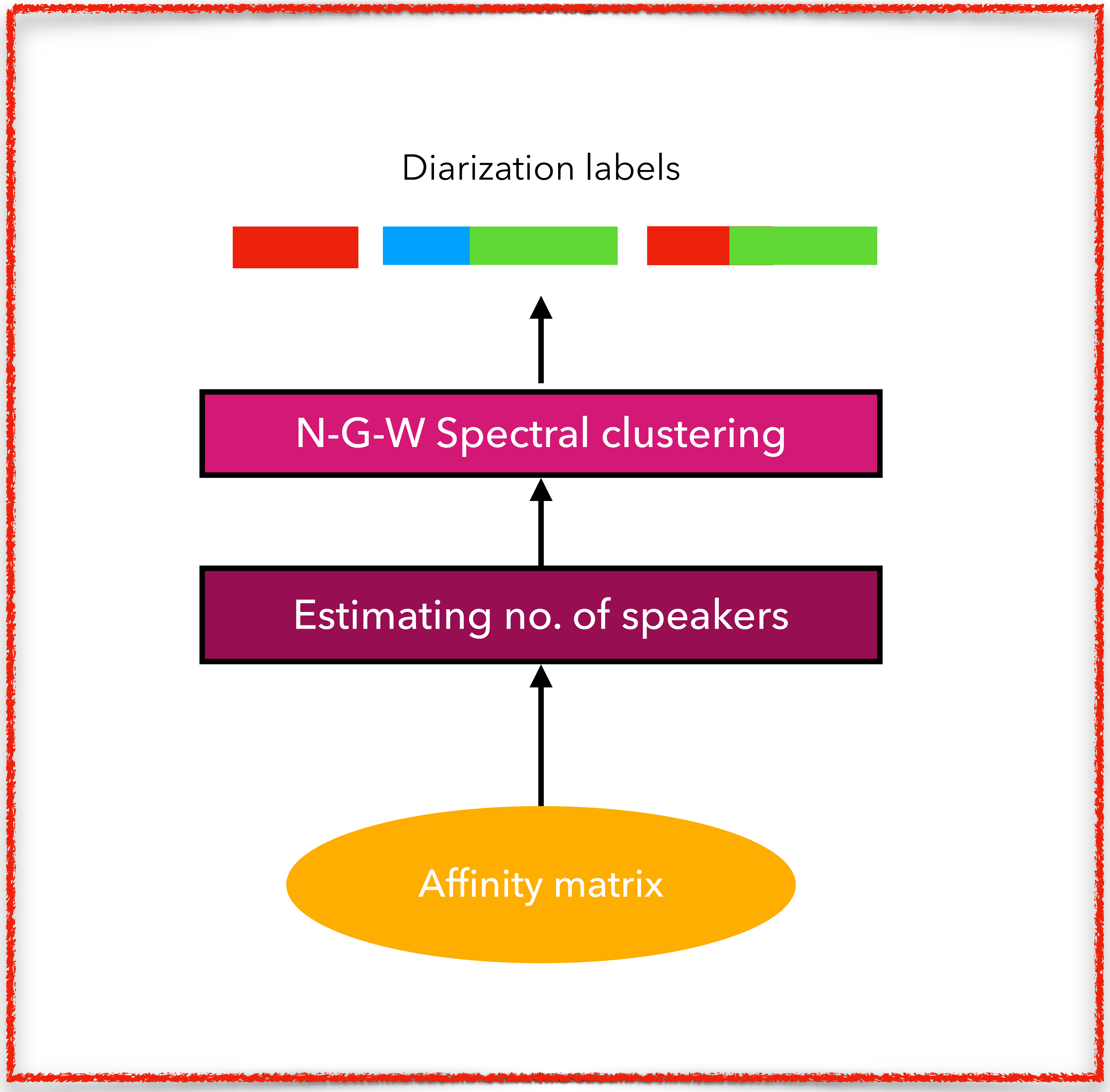
# Overlap-aware spectral clustering

## Overview of differences

**Regular spectral clustering**

**(Ng-Jordan-Weiss algorithm):**

- Estimate number of speakers (say, $K$)
- Compute Laplacian $L$ of affinity matrix
- Apply K-means clustering on first $K$ eigenvectors of $L$




Andrew Y. Ng, Michael I. Jordan, and Yair Weiss, "On spectral clustering: Analysis and an algorithm," NIPS, 2001

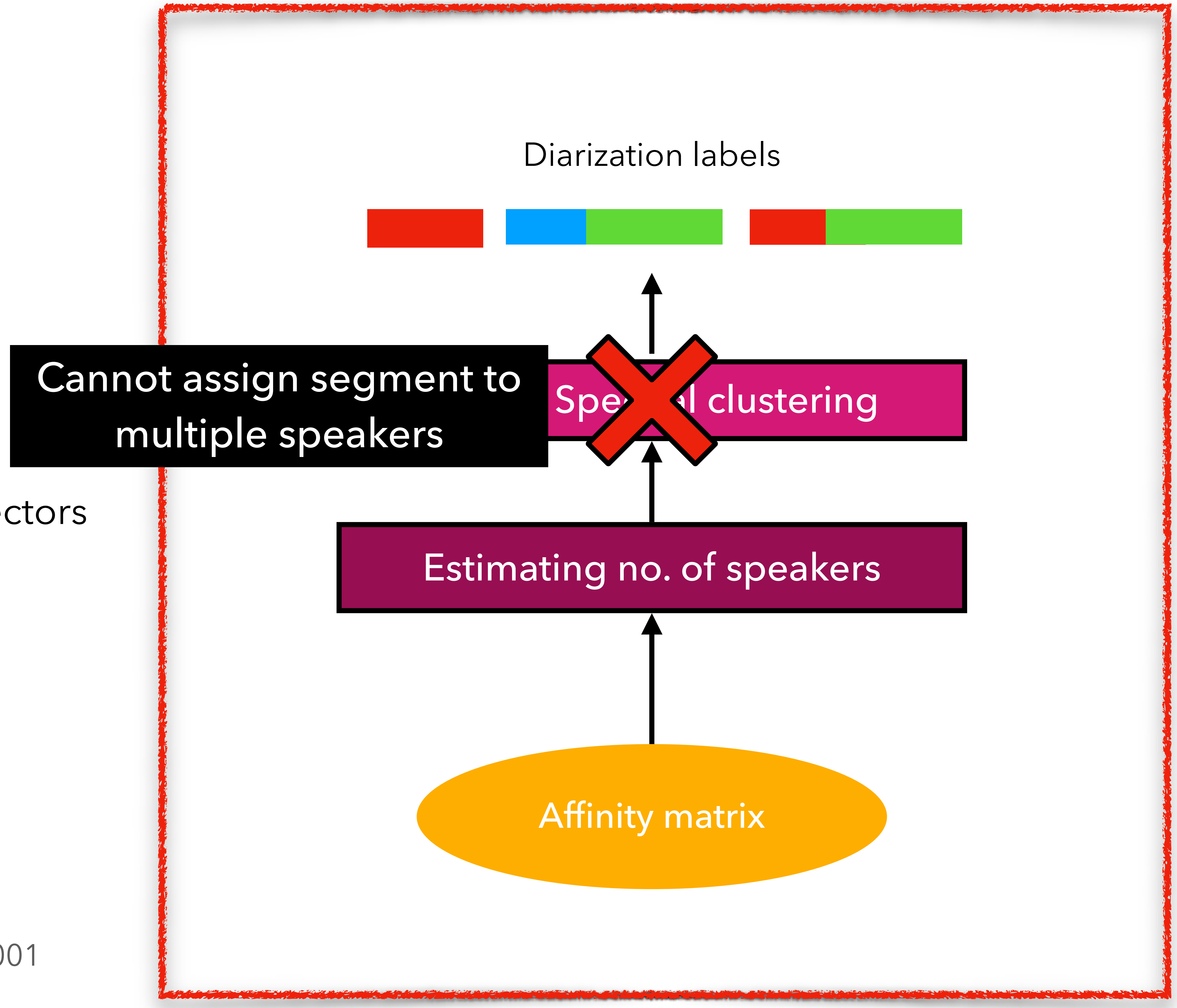
# Overlap-aware spectral clustering

## Overview of differences

**Regular spectral clustering**

**(Ng-Jordan-Weiss algorithm):**

- Estimate number of speakers (say, *K*)
- Compute Laplacian *L* of affinity matrix
- Apply K-means clustering on first *K* eigenvectors of *L*




Andrew Y. Ng, Michael I. Jordan, and Yair Weiss, “On spectral clustering: Analysis and an algorithm,” NIPS, 2001

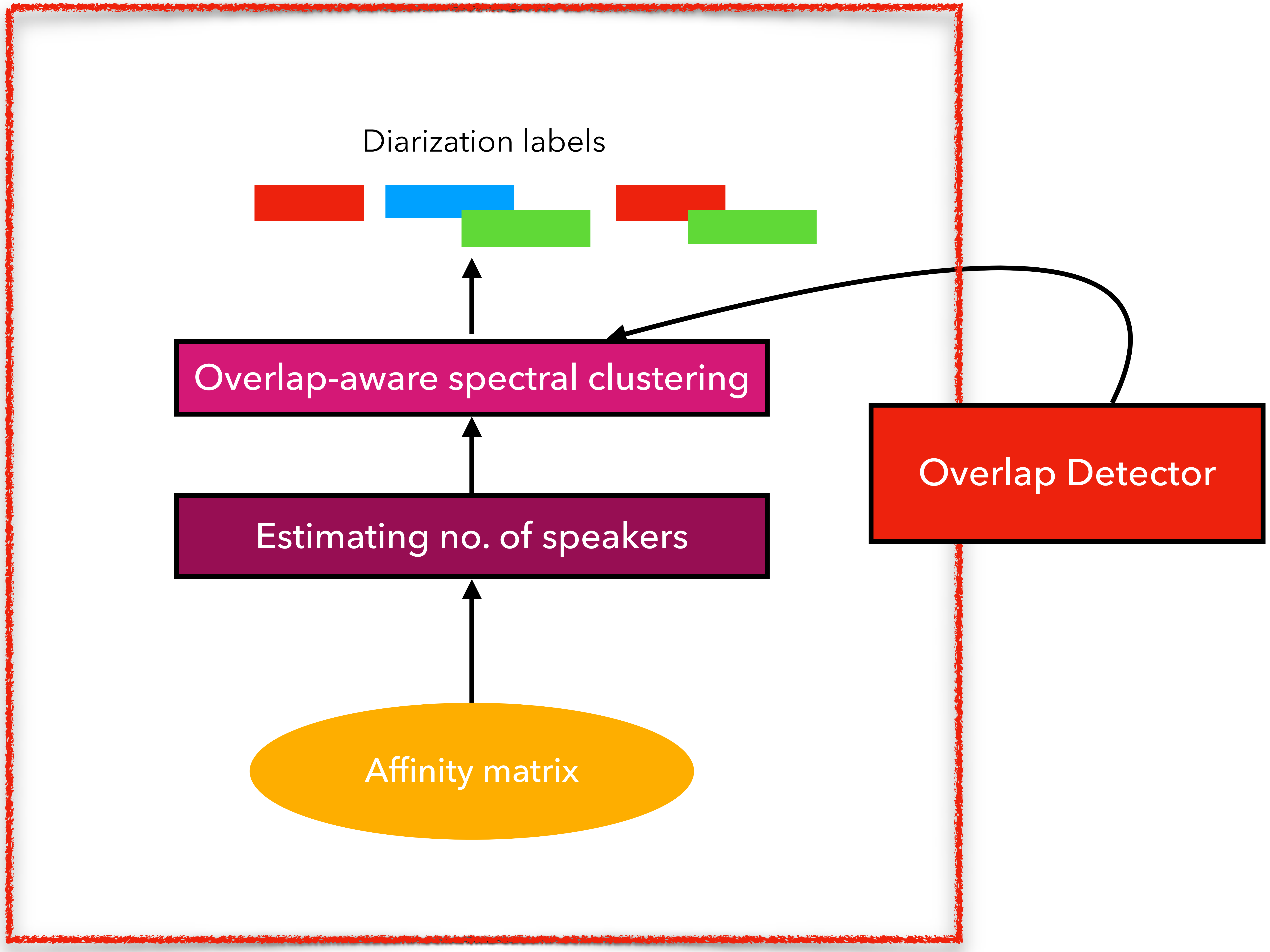
# Overlap-aware spectral clustering

## Overview of differences

**Alternative formulation:**

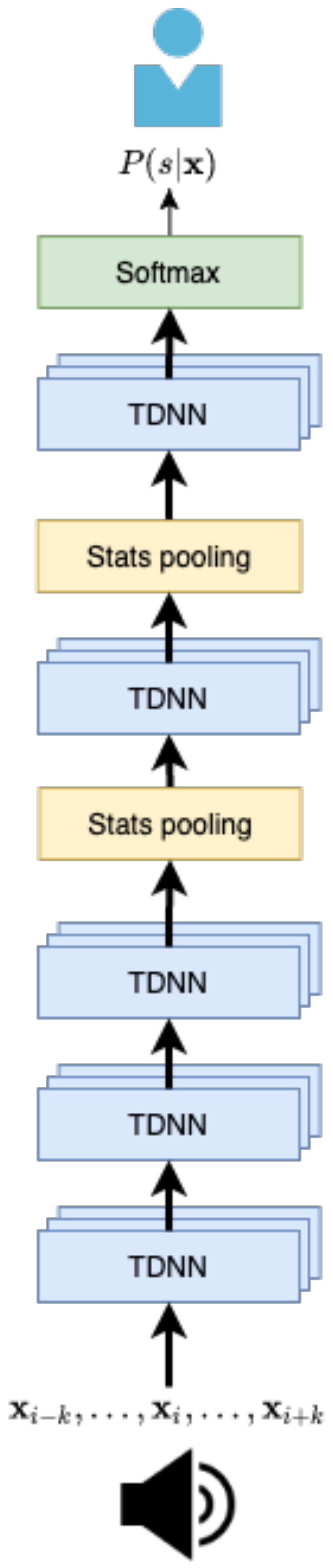
**multi-class spectral clustering**

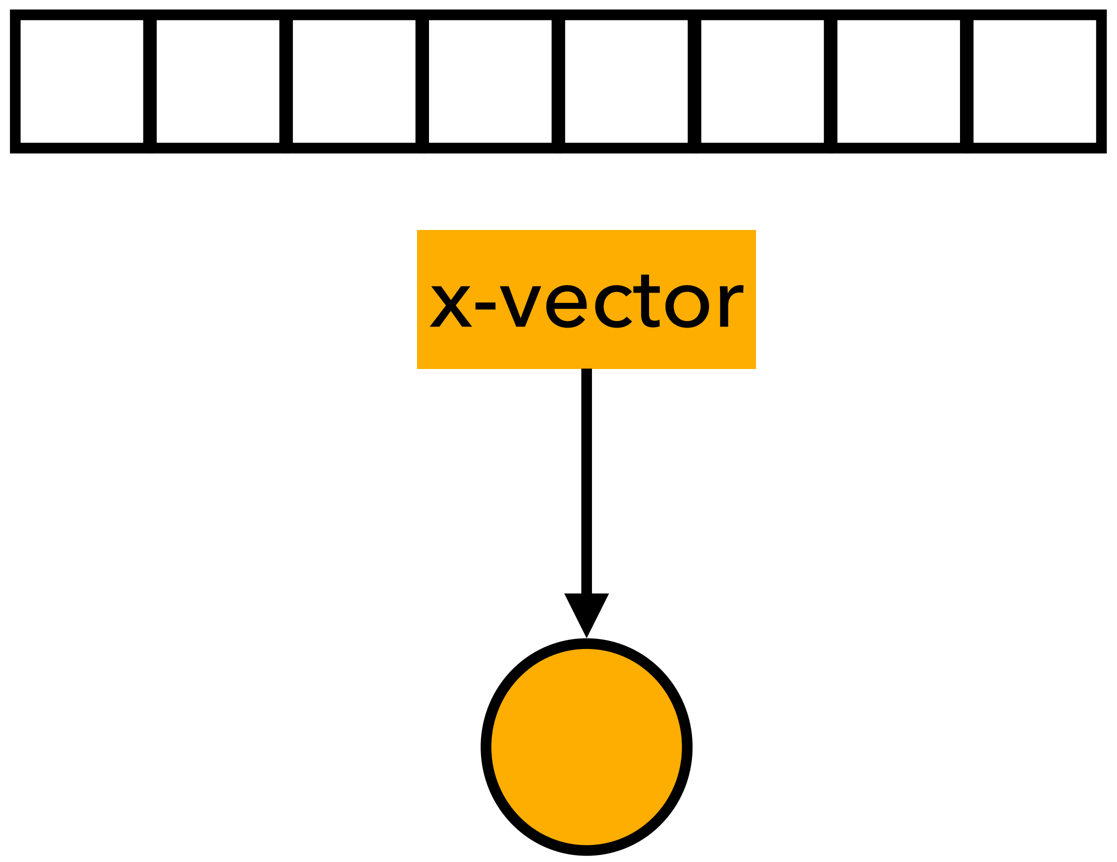
Yu, S., & Shi, J. Multiclass spectral clustering. ICCV 2003.

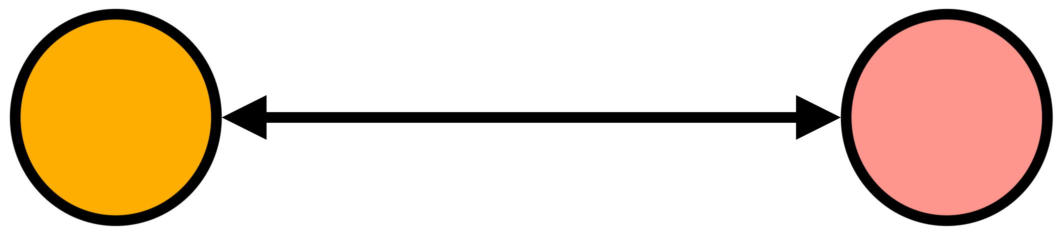


# New formulation for spectral clustering

## The basic clustering problem: a graph view



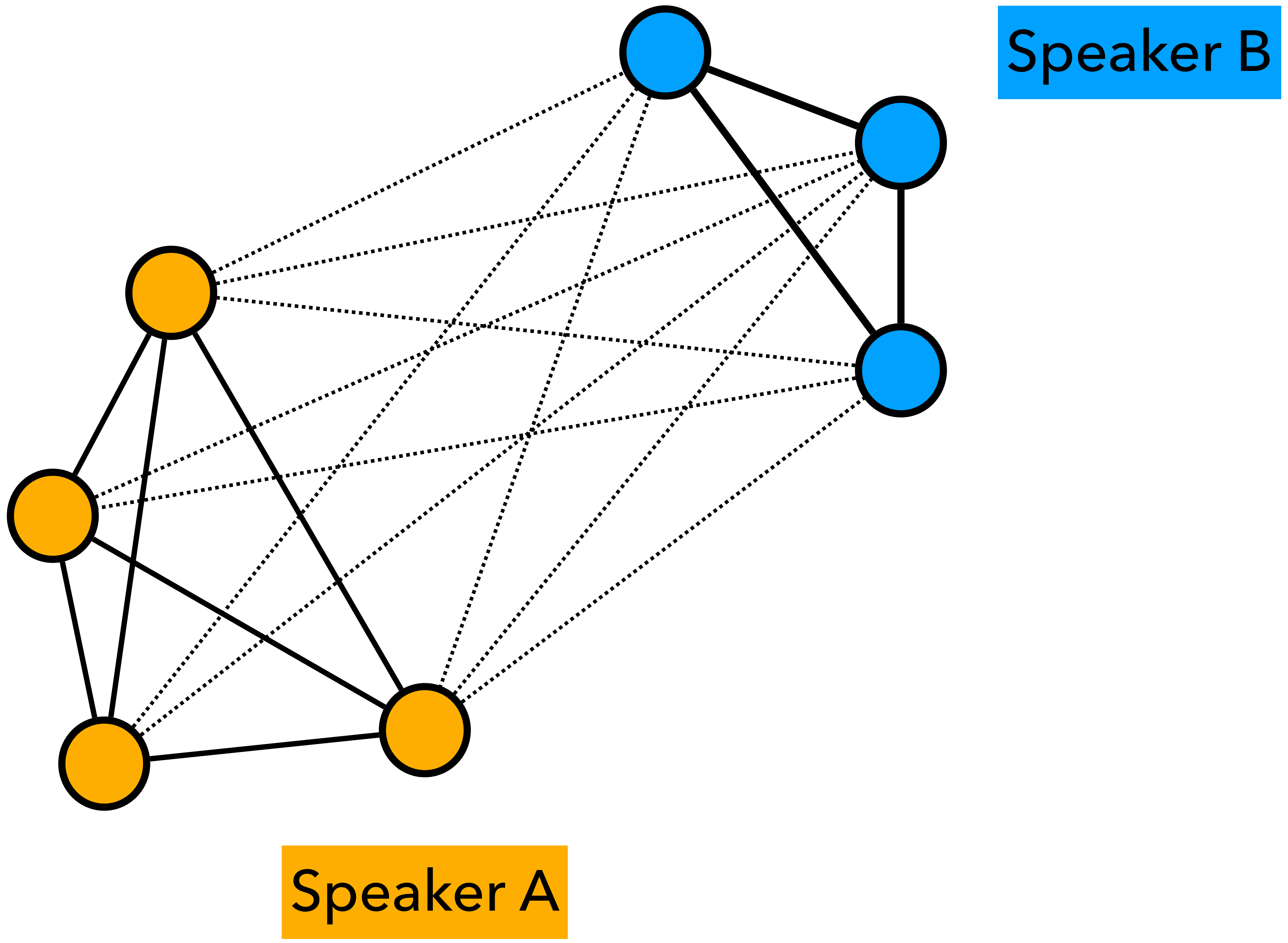



**Cosine similarity**

Snyder, D., et al. (2018). X-Vectors: Robust DNN Embeddings for Speaker Recognition. *2018 IEEE ICASSP*.

# New formulation for spectral clustering

## The basic clustering problem: a graph view

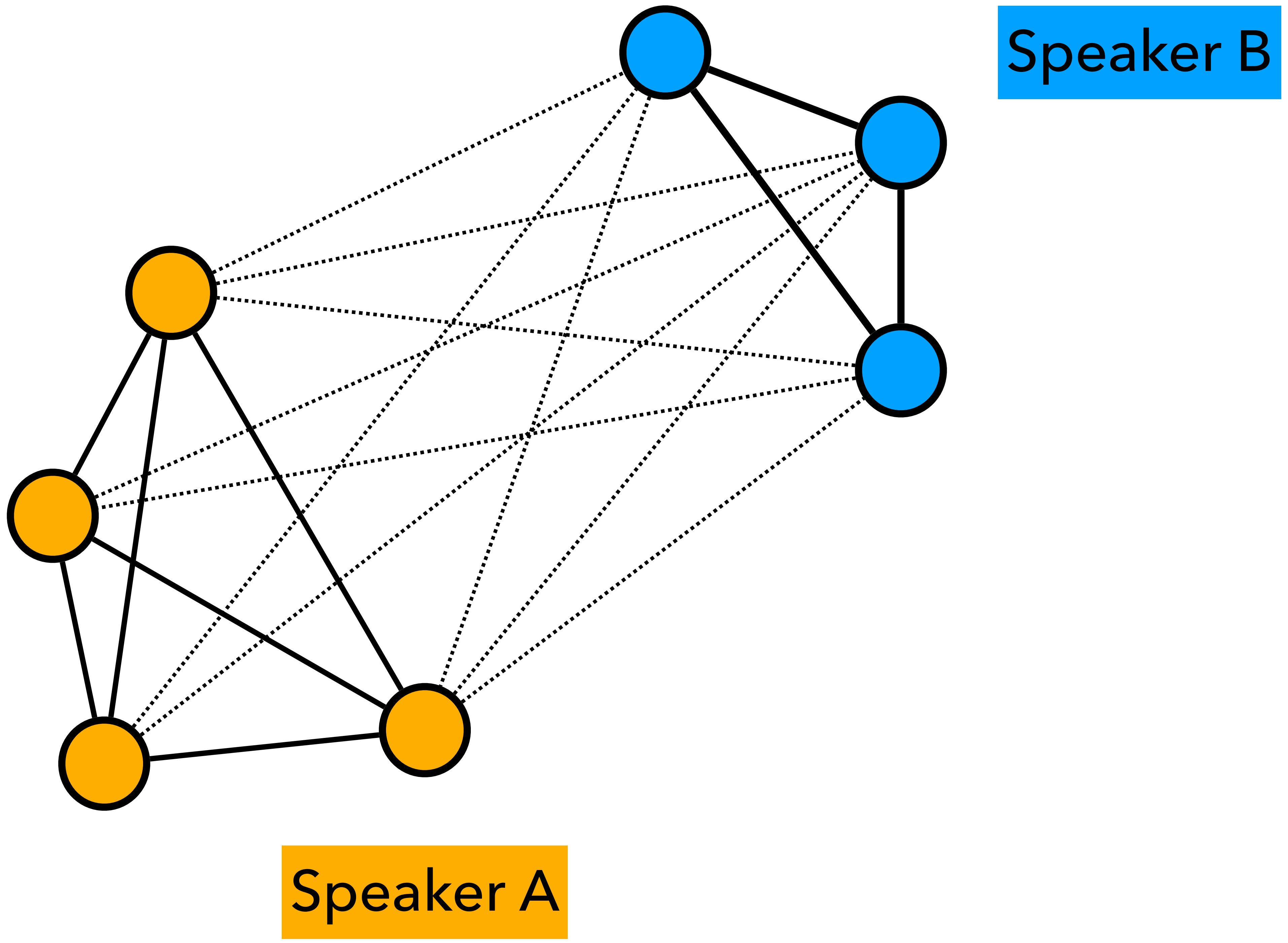



**Edge weights within a group**

**Edge weights across groups**

# New formulation for spectral clustering

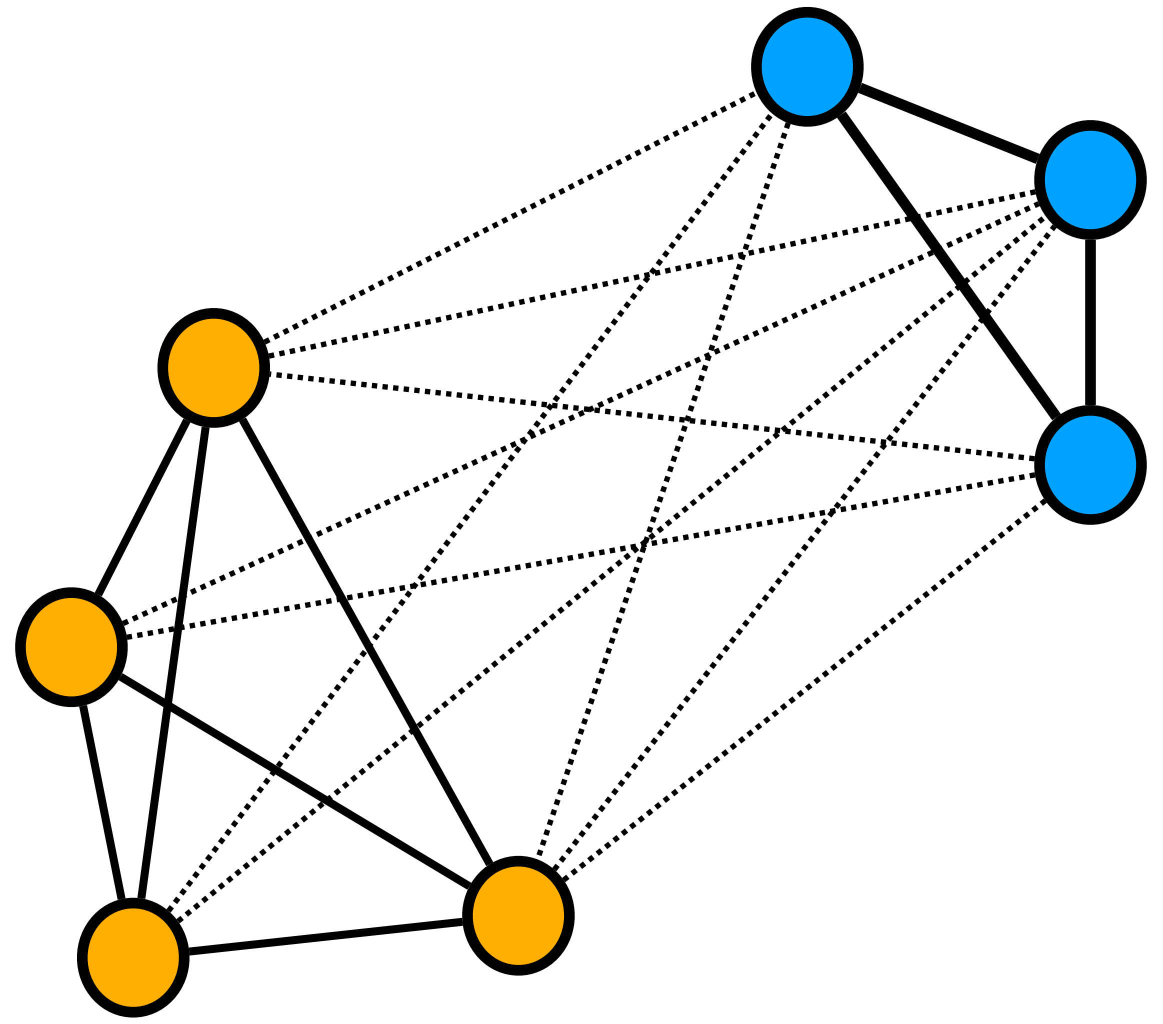
## The basic clustering problem: a graph view




$$\textit{maximize} \quad \frac{\textbf{Edge weights within a group}}{\textbf{Edge weights across groups}}$$

# New formulation for spectral clustering

## The basic clustering problem: a graph view



***maximize*** $\dfrac{\textbf{Edge weights within a group}}{\textbf{Edge weights across groups}}$

$$\text{maximize} \quad \epsilon(X) = \frac{1}{K}\sum_{k=1}^{K}\frac{X_k^T\mathbf{A}X_k}{X_k^T\mathbf{D}X_k}$$

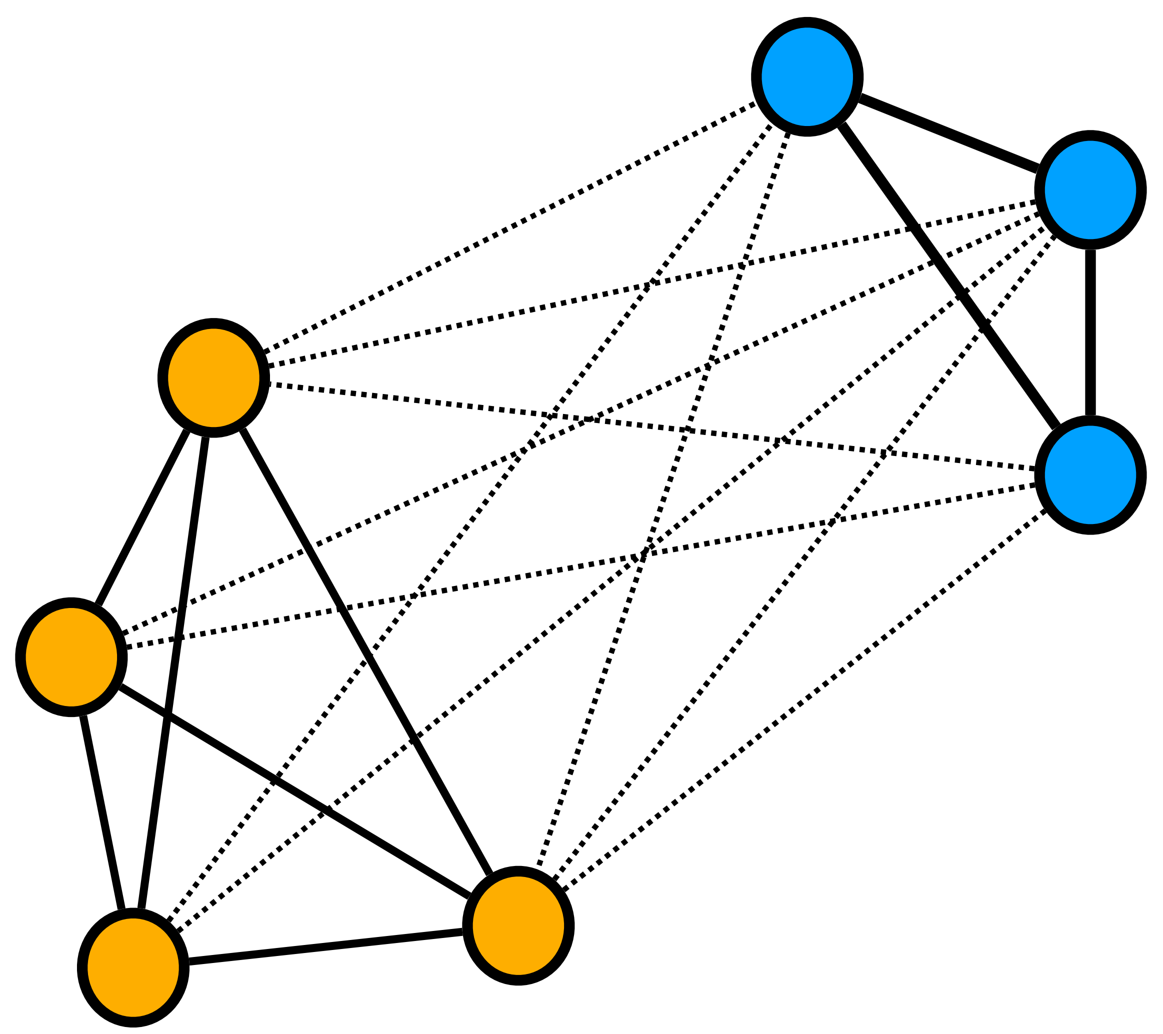
$$\text{subject to} \quad X \in \{0,1\}^{N\times K},$$

$$X\mathbf{1}_K = \mathbf{1}_N\,.$$

**K** speakers, **N** segments

# New formulation for spectral clustering

## The basic clustering problem: a graph view



***maximize*** $\dfrac{\textbf{Edge weights within a group}}{\textbf{Edge weights across groups}}$

$$\text{maximize} \quad \epsilon(X) = \frac{1}{K}\sum_{k=1}^{K}\frac{X_k^T \mathbf{A} X_k}{X_k^T \mathbf{D} X_k}$$

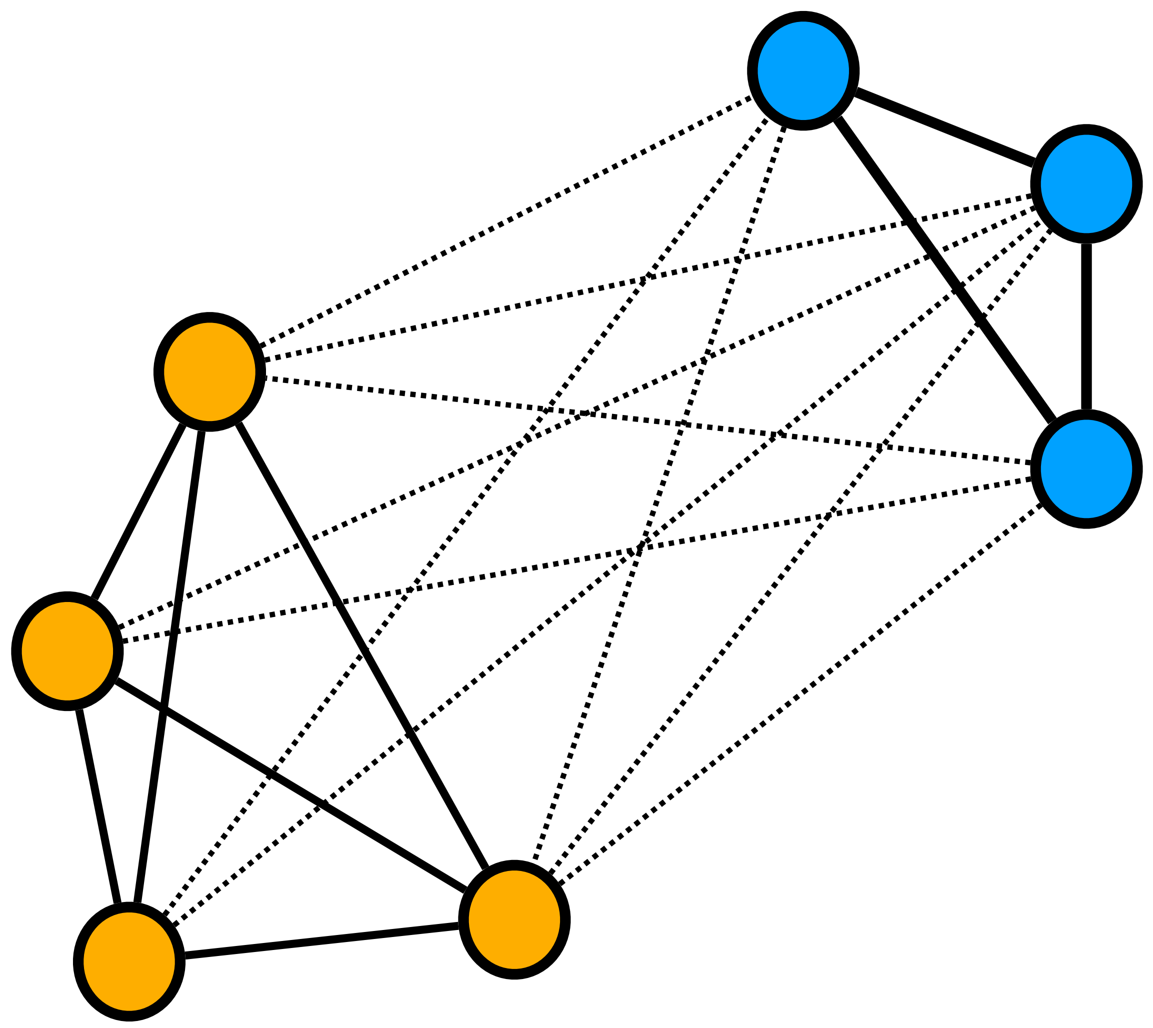
$$\text{subject to} \quad X \in \{0,1\}^{N\times K}, \quad X\mathbf{1}_K = \mathbf{1}_N.$$

Affinity

Diagonal matrix containing degree of nodes

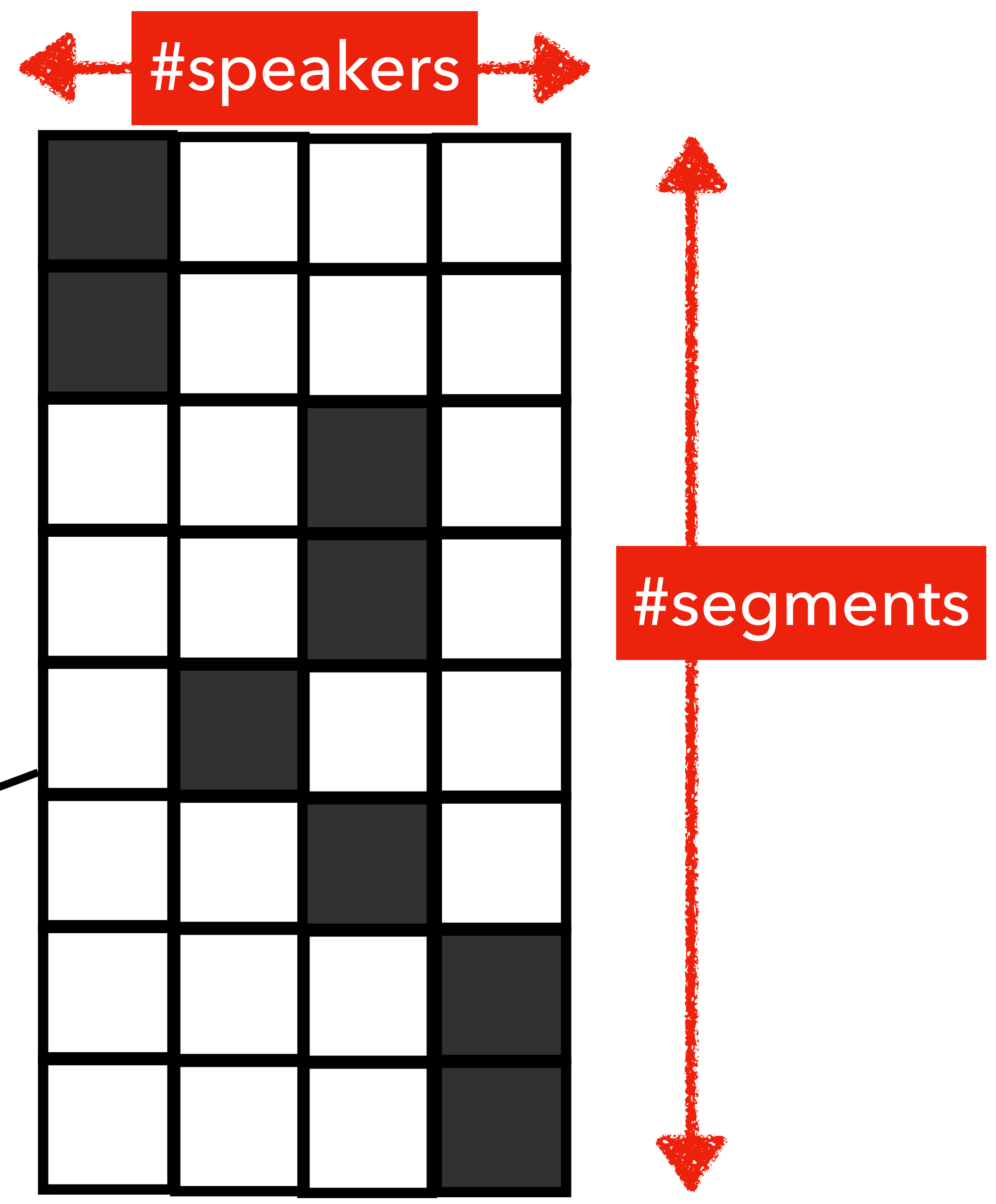
# New formulation for spectral clustering

## The basic clustering problem: a graph view



$$
\begin{aligned}
\text{maximize} \quad & \epsilon(X) = \frac{1}{K} \sum_{k=1}^{K} \frac{X_k^T \mathbf{A} X_k}{X_k^T \mathbf{D} X_k} \\
\text{subject to} \quad & X \in \{0,1\}^{N \times K}, \\
& X \mathbf{1}_K = \mathbf{1}_N .
\end{aligned}
$$

Final cluster assignment matrix



# New formulation for spectral clustering

## This problem is NP-hard!

$$
\begin{aligned}
\text{maximize} \quad & \epsilon(X) = \frac{1}{K} \sum_{k=1}^{K} \frac{X_k^T \mathbf{A} X_k}{X_k^T \mathbf{D} X_k} \\
\text{subject to} \quad & \cancel{X \in \{0,1\}^{N \times K}}, \\
& \cancel{X \mathbf{1}_K = \mathbf{1}_N}.
\end{aligned}
$$

**Remove the discrete constraints** to make the problem solvable

# New formulation for spectral clustering

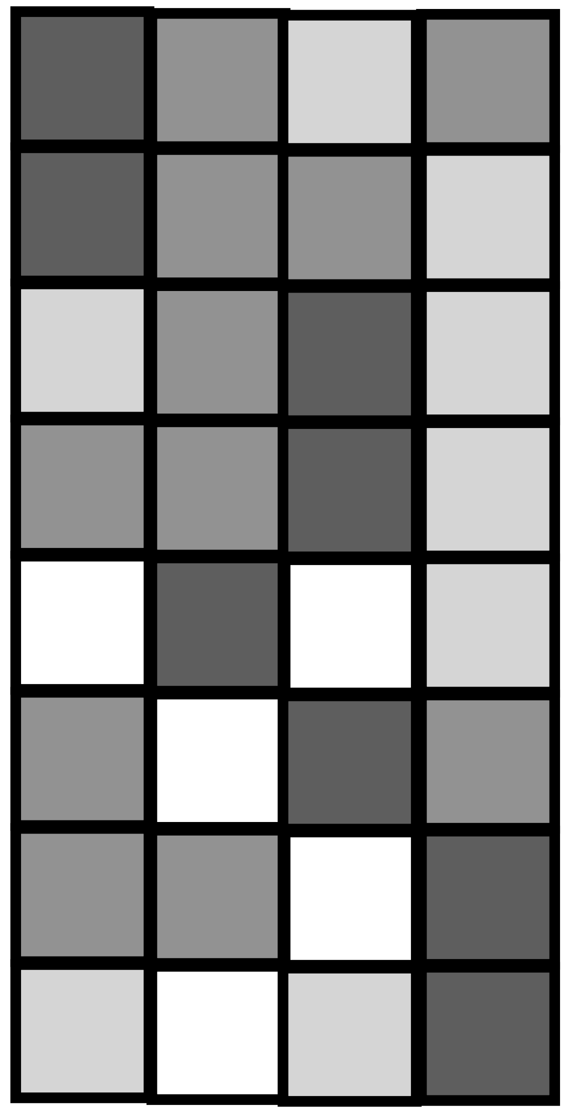
## Relaxed problem has a set of solutions

$$\text{maximize} \quad \epsilon(X) = \frac{1}{K}\sum_{k=1}^{K} \frac{X_k^T \mathbf{A} X_k}{X_k^T \mathbf{D} X_k}$$

$$\text{subject to} \quad \cancel{X \in \{0,1\}^{N\times K},}$$

$$\cancel{X\mathbf{1}_K = \mathbf{1}_N.}$$

**Taking the Eigen-decomposition of $D^{-1}A$**

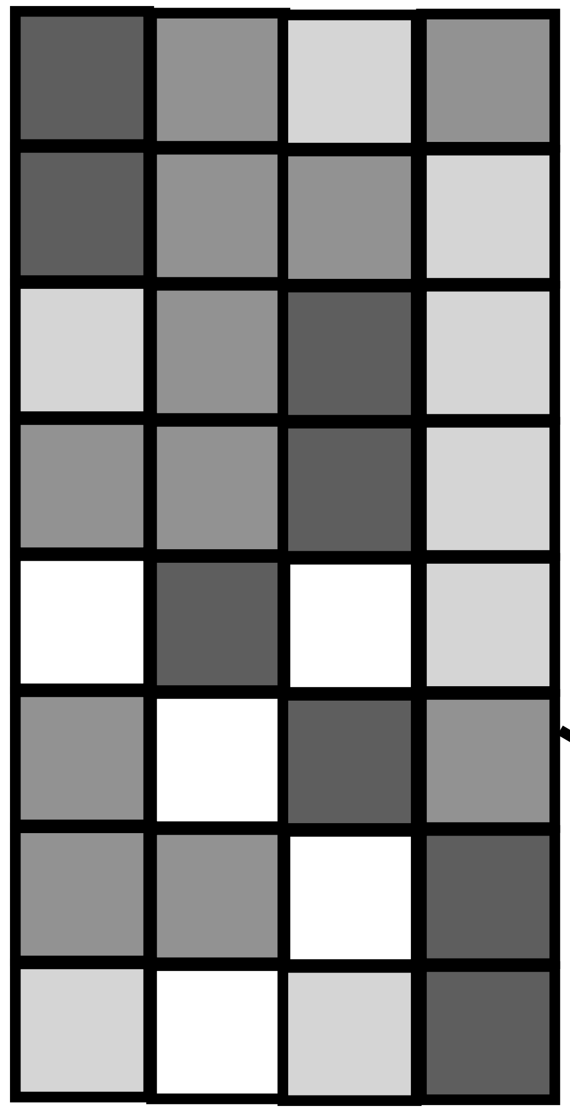


and its orthonormal transforms

**Set of solutions** to the **relaxed** problem

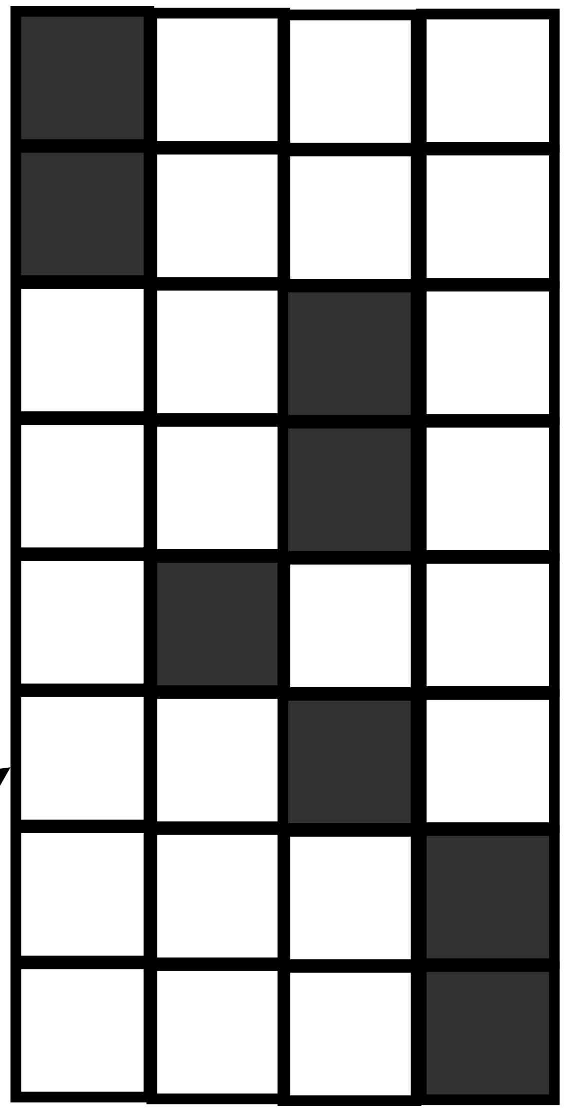
# New formulation for spectral clustering

## Now we need to discretize this solution!



and its orthonormal transforms

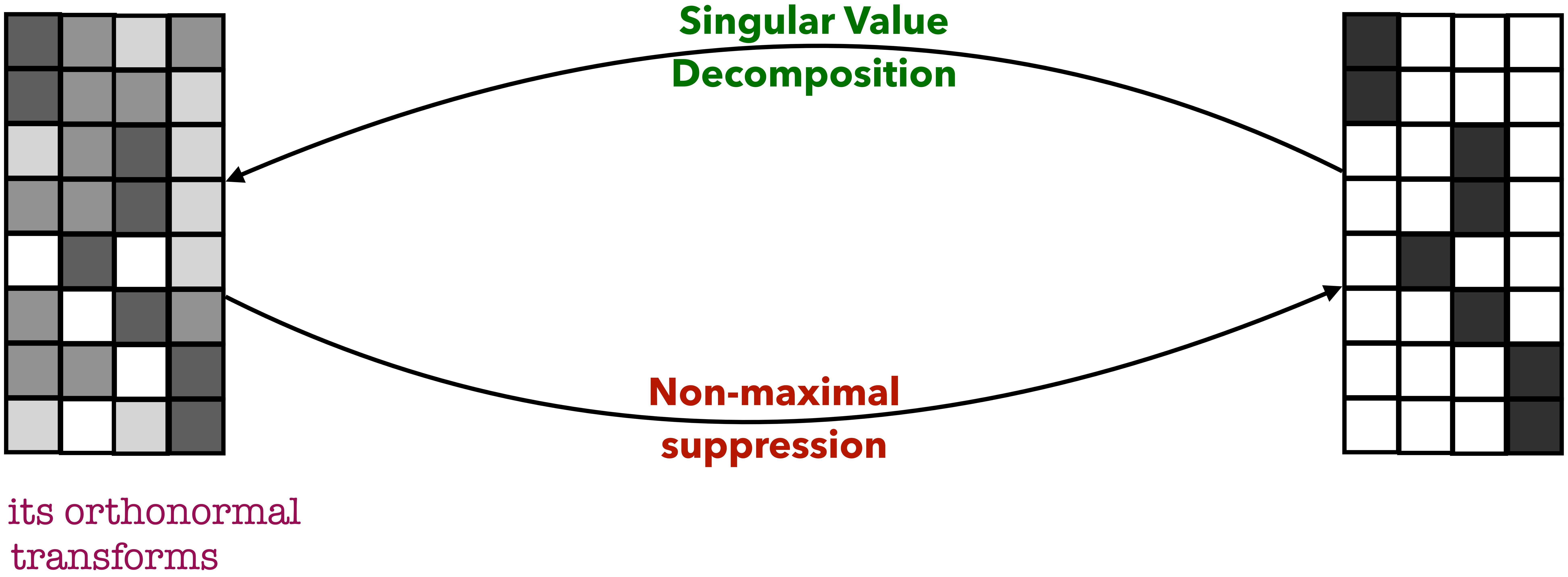
$$\text{subject to} \quad X \in \{0,1\}^{N \times K}, \quad X\mathbf{1}_K = \mathbf{1}_N .$$



Find a matrix which is **discrete** and also close to any one of the **orthonormal transformations** of the relaxed solution

# New formulation for spectral clustering

## Now we need to **discretize** this solution!

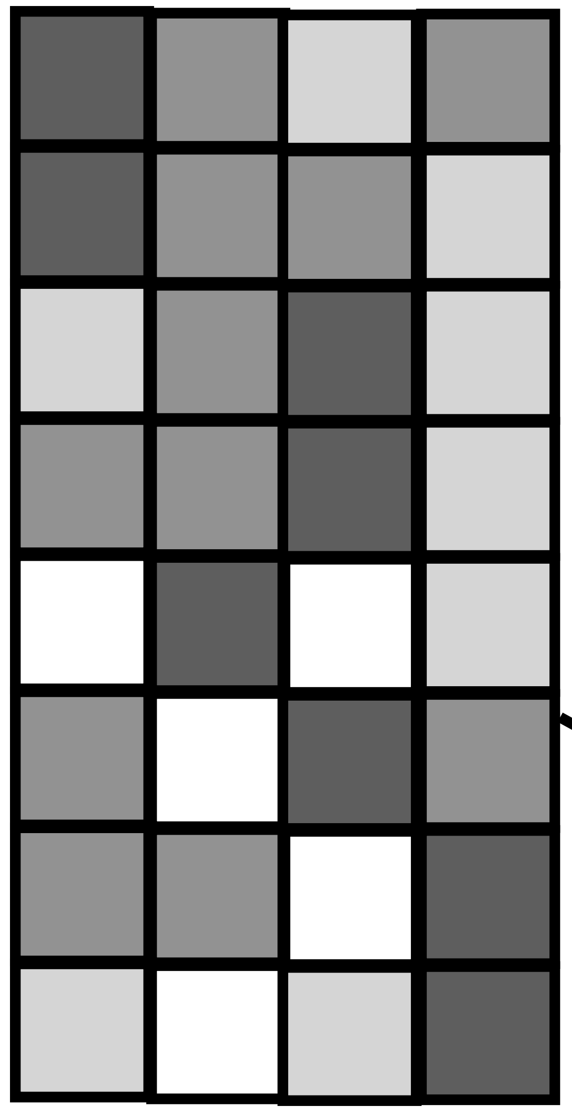



**Iterate until convergence**

# Let us now make it overlap-aware

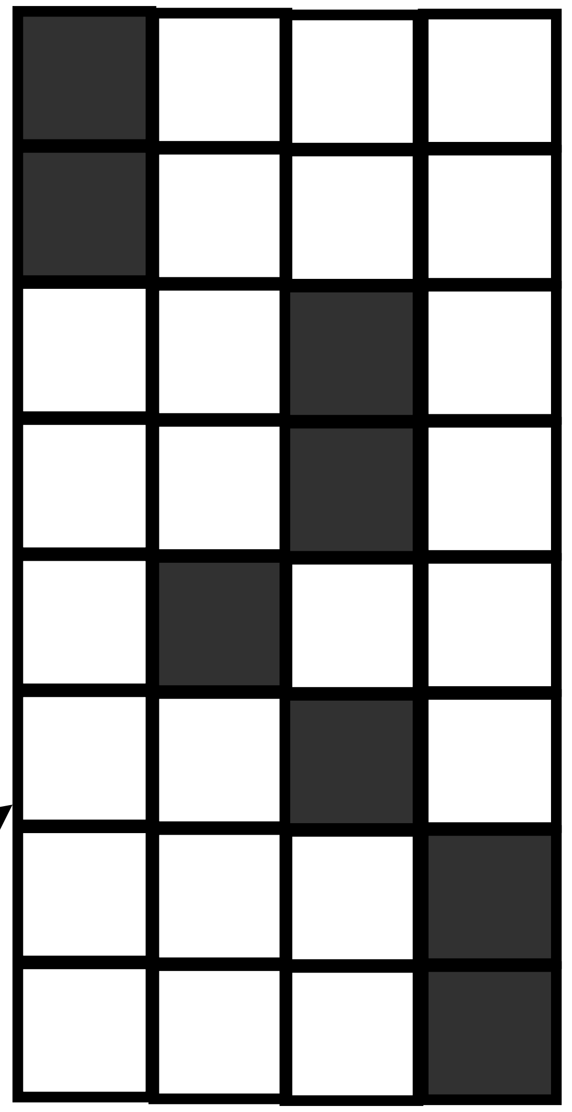
## Suppose we have $v_{OL}$

Overlap Detector



and its orthonormal transforms

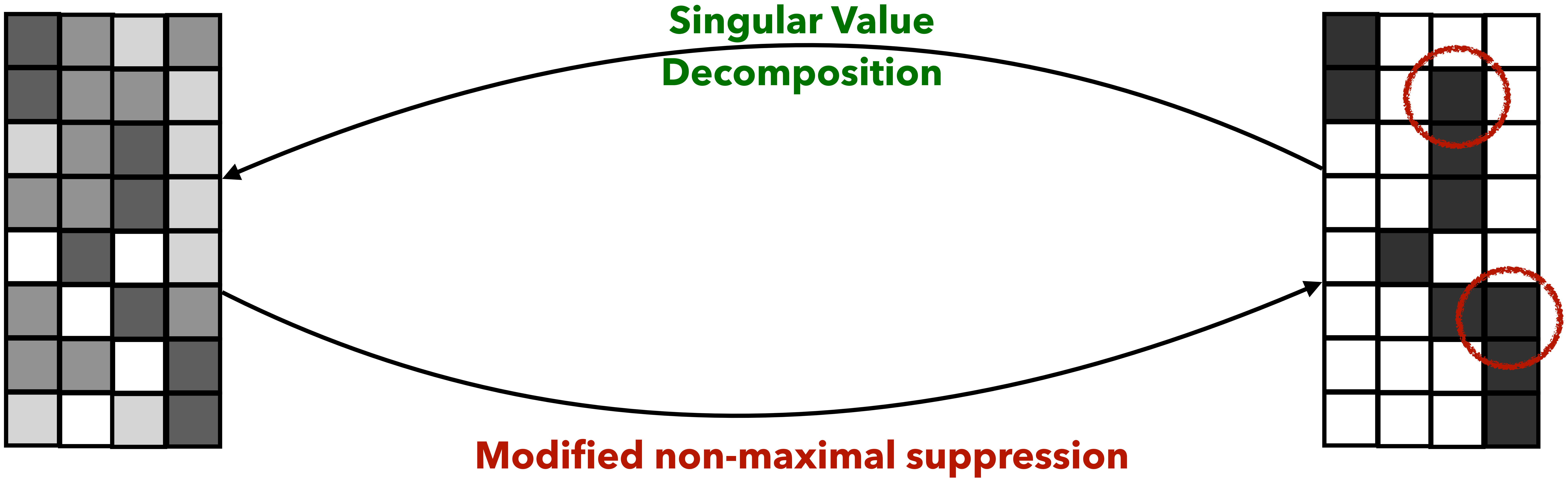
$$\text{subject to} \quad X \in \{0,1\}^{N\times K}, \quad X\mathbf{1}_K = \mathbf{1}_N + \boldsymbol{v}_{OL}.$$



**Discrete constraint is modified to include overlap detector output**

# Let us now make it overlap-aware

## Modify non-maximal suppression to pick top 2 speakers




and its orthonormal transforms

**Iterate until convergence**

# Results on AMI Mix-Headset eval

## 12.0% relative improvement over spectral clustering baseline

| System | DER |
|---|---|
| **Spectral clustering** | 26.9 |
| **AHC** | 28.3 |
| **VBx** | 26.2 |
| **Overlap-aware SC** | **24.0** |


Park et al., "Auto-tuning spectral clustering for speaker diarization using normalized maximum eigengap," IEEE Signal Processing Letters, 2020.

Garcia-Romero et al., "Speaker diarization using deep neural network embeddings," ICASSP 2017.

Dîez et al., "Speaker diarization based on Bayesian HMM with eigenvoice priors," Odyssey 2018.


AMI data contains **4-speaker meetings**

# Results on AMI Mix-Headset eval

## Comparable with other overlap-aware diarization methods

| System | DER |
|---|---|
| **VB-based overlap assignment** | 23.8 |
| **Region proposal networks** | 25.5 |
| **Overlap-aware SC** | **24.0** |

Bullock, et al., "Overlap-aware diarization: resegmentation using neural end-toend overlapped speech detection," ICASSP 2020.

Huang et al., "Speaker diarization with region proposal network," ICASSP 2020.

Does not require **matching training data** or **initialization** with other diarization systems.

# Results: DER breakdown on AMI eval

| System | Missed speech | False alarm | Speaker conf. | DER |
|---|---|---|---|---|
| **AHC/PLDA** | 19.9 | 0.0 | 8.4 | 26.9 |
| **Spectral/cosine** | 19.9 | 0.0 | 7.0 | 28.3 |
| **VBx** | 19.9 | 0.0 | 6.3 | 26.2 |
| **VB-based overlap assignment** | 13.0 | 3.6 | 7.2 | 23.8 |
| **RPN** | 9.5 | 7.7 | 8.3 | 25.5 |
| **Overlap-aware SC** | 11.3 | 2.2 | 10.5 | 24.0 |

# Results: DER breakdown on AMI eval

## Missed speech decreases significantly

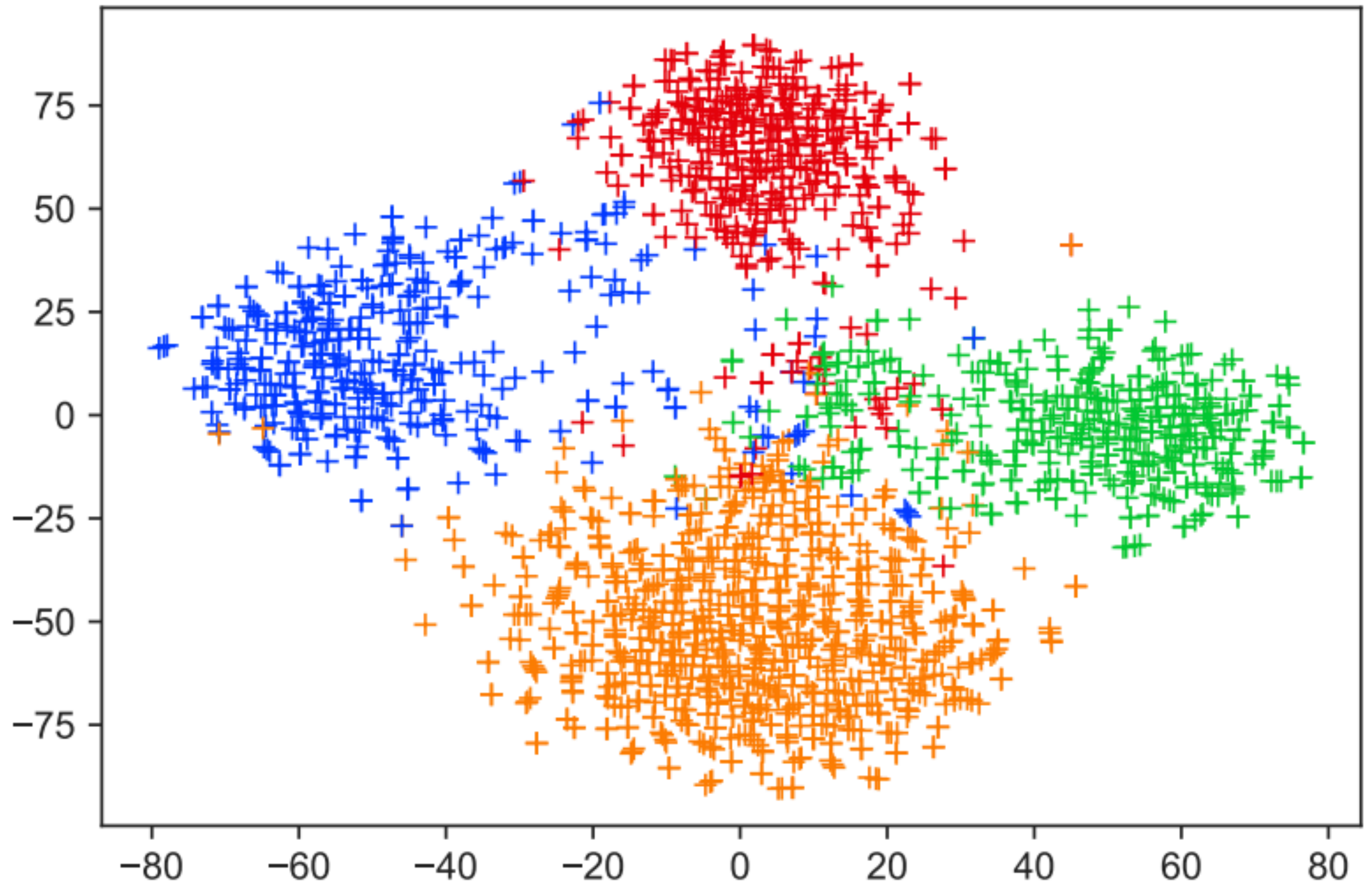
| System | Missed speech | False alarm | Speaker conf. | DER |
|---|---|---|---|---|
| **AHC/PLDA** | 19.9 | 0.0 | 8.4 | 26.9 |
| **Spectral/cosine** | 19.9 | 0.0 | 7.0 | 28.3 |
| **VBx** | 19.9 | 0.0 | 6.3 | 26.2 |
| **VB-based overlap assignment** | 13.0 | 3.6 | 7.2 | 23.8 |
| **RPN** | 9.5 | 7.7 | 8.3 | 25.5 |
| **Overlap-aware SC** | 11.3 | 2.2 | 10.5 | 24.0 |

# Results: DER breakdown on AMI eval

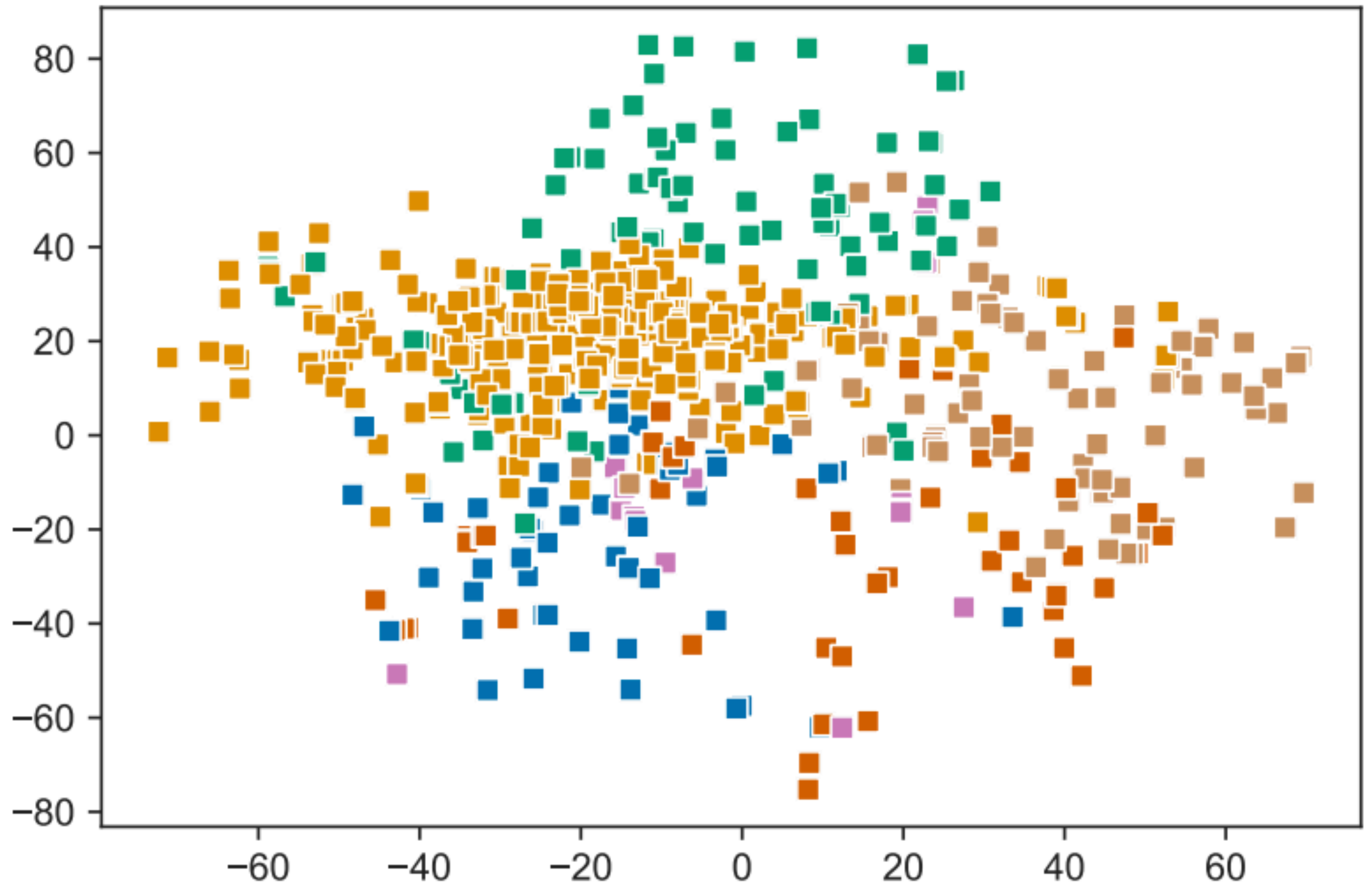
## Speaker confusion increases

| System | Missed speech | False alarm | Speaker conf. | DER |
|---|---|---|---|---|
| **AHC/PLDA** | 19.9 | 0.0 | 8.4 | 26.9 |
| **Spectral/cosine** | 19.9 | 0.0 | 7.0 | 28.3 |
| **VBx** | 19.9 | 0.0 | 6.3 | 26.2 |
| **VB-based overlap assignment** | 13.0 | 3.6 | 7.2 | 23.8 |
| **RPN** | 9.5 | 7.7 | 8.3 | 25.5 |
| **Overlap-aware SC** | 11.3 | 2.2 | 10.5 | 24.0 |

# Need more robust x-vector extractors




Non-overlapping segments




Overlapping segments

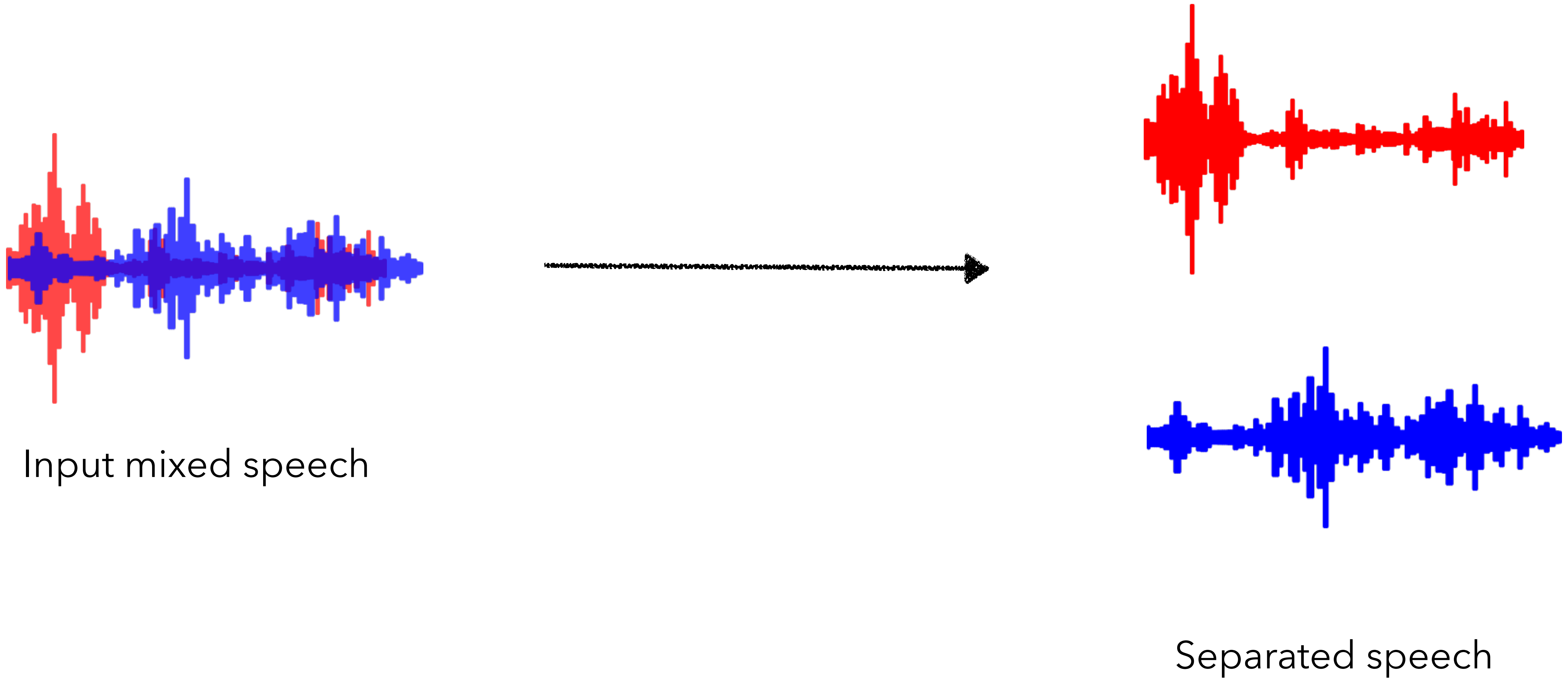
**T-SNE plot** of x-vector embeddings

# Continuous Speech Separation

# What is continuous speech separation?

## Motivation

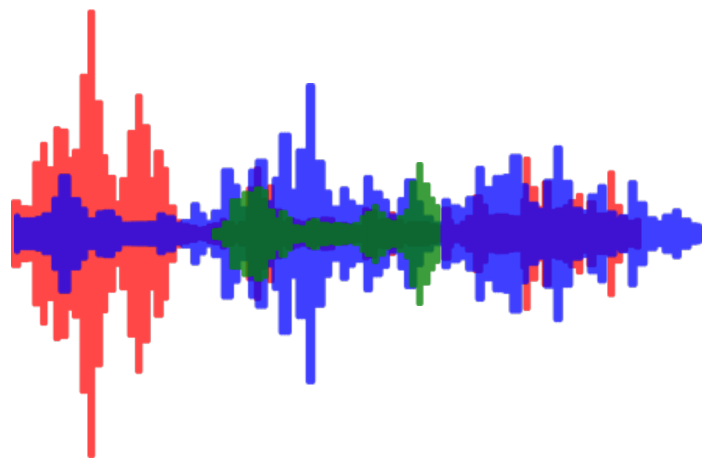
- Speech separation using neural networks works well for fixed number of speakers, e.g., separating short 2-speaker mixtures



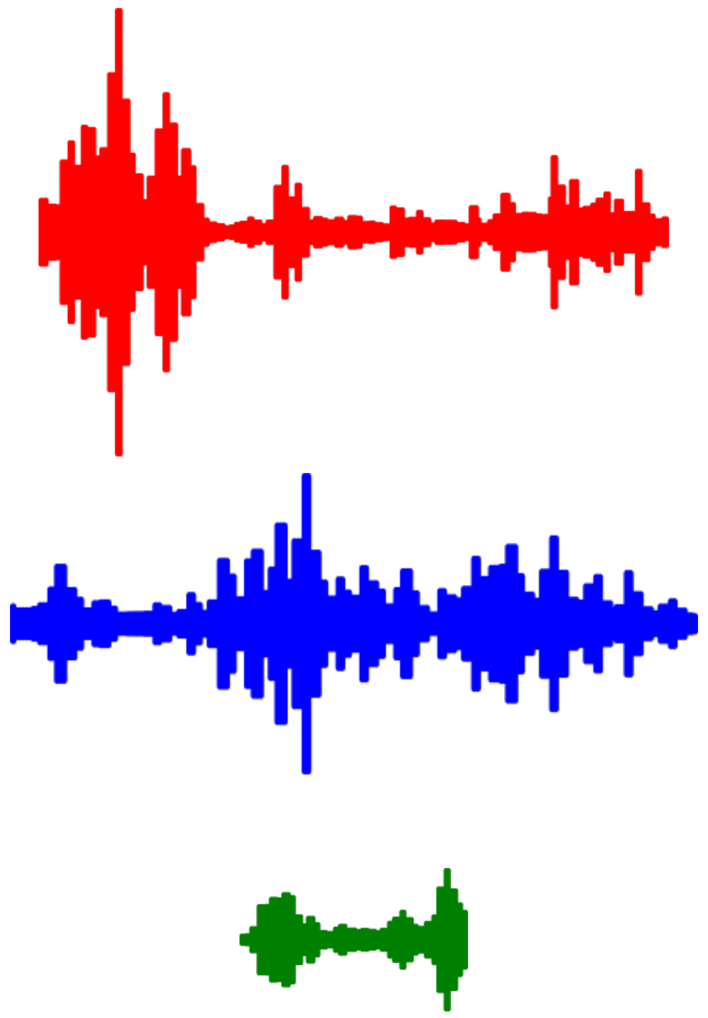
# What is continuous speech separation?

## Motivation

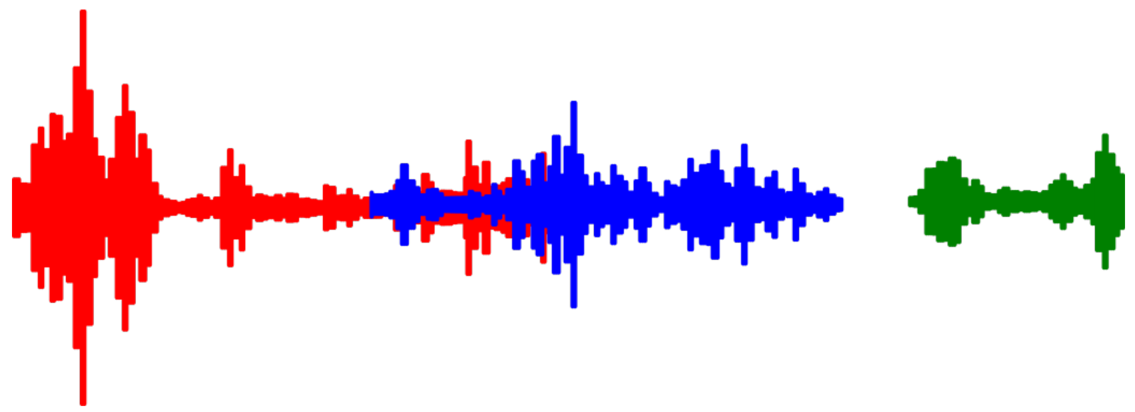
- But what about arbitrary number of speakers? **Problem:** neural networks are trained with fixed number of outputs
- Or long-form recordings? **Problem:** OOM



Mixed speech with 3 speakers



Separated speech

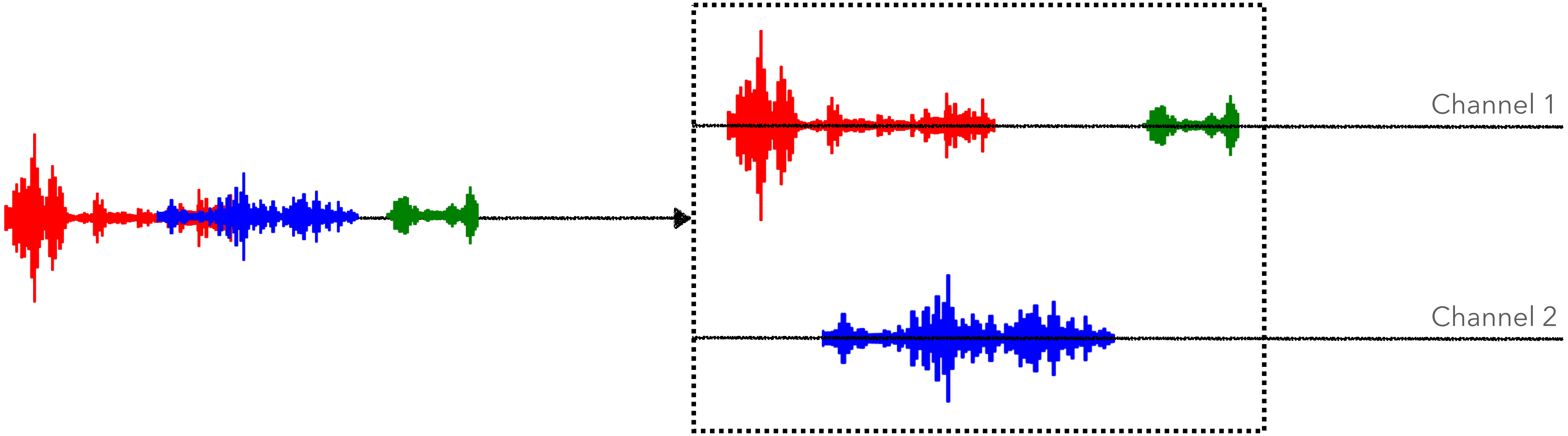


Long recording containing 3 speakers

# What is continuous speech separation?

## Idea: separate small chunks

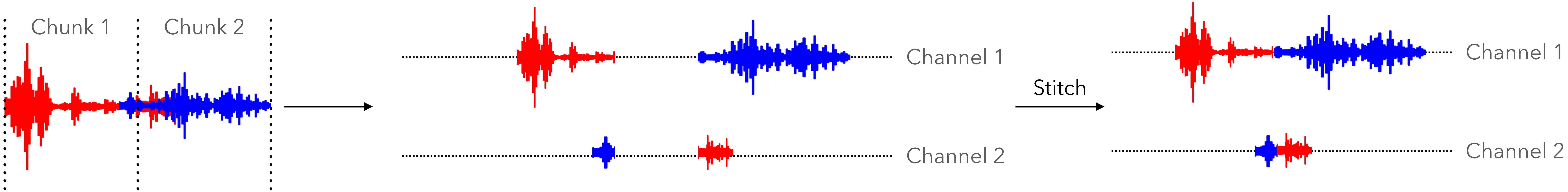
- Assumption: A small segment (say 2-3 seconds) will contain at most 2 speakers
- Separate small chunks into fixed number of outputs and **stitch**



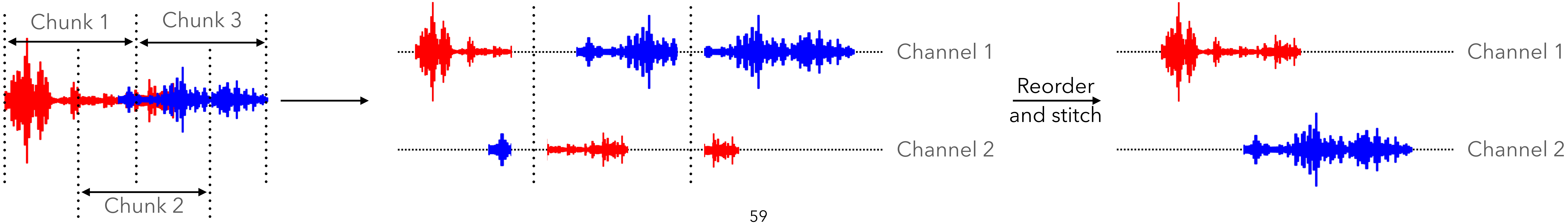
# What is continuous speech separation?

## Caveat: permutation problem between chunks

- **Problem**: Output order may change across chunks, causing discontinuity




- **Solution**: use "overlapping" chunks and reorder masks based on shared portion to minimize cross-entropy



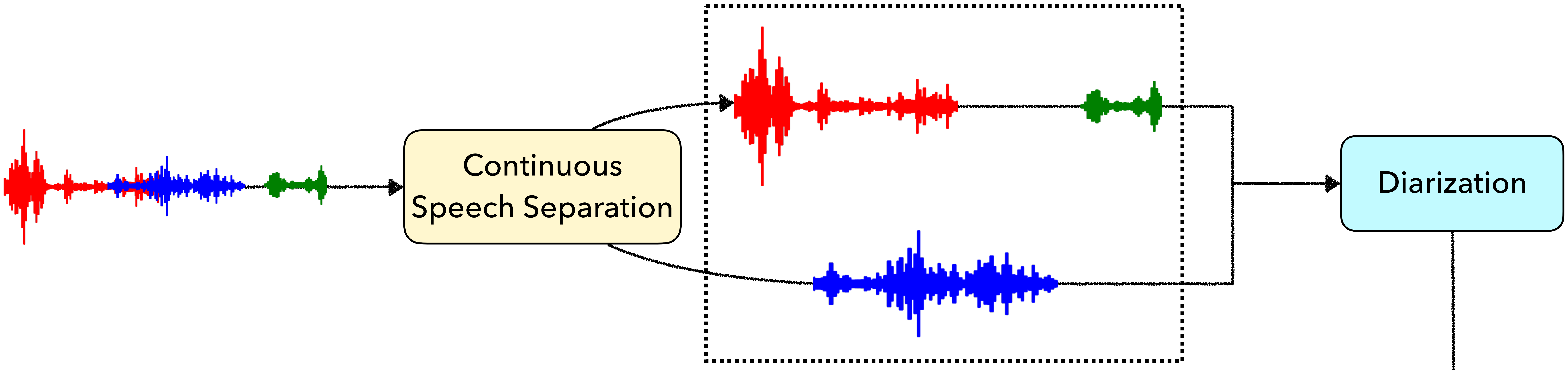
# CSS-based diarization

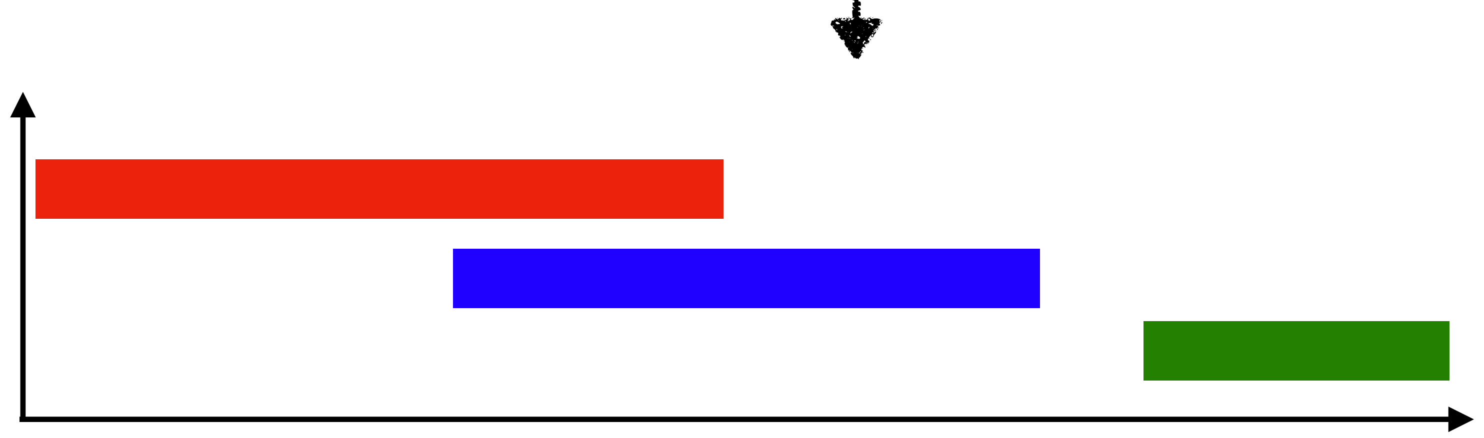
# Motivation

## A different paradigm for overlap-aware diarization

1. It is hard to train a good overlap detector
   - Data sparsity issue
   - Need frame-level alignments
2. Speaker embedding extractors may not produce good representations of overlapping segments
3. For CSS-based systems, we already have access to separated audio streams

# CSS-based diarization



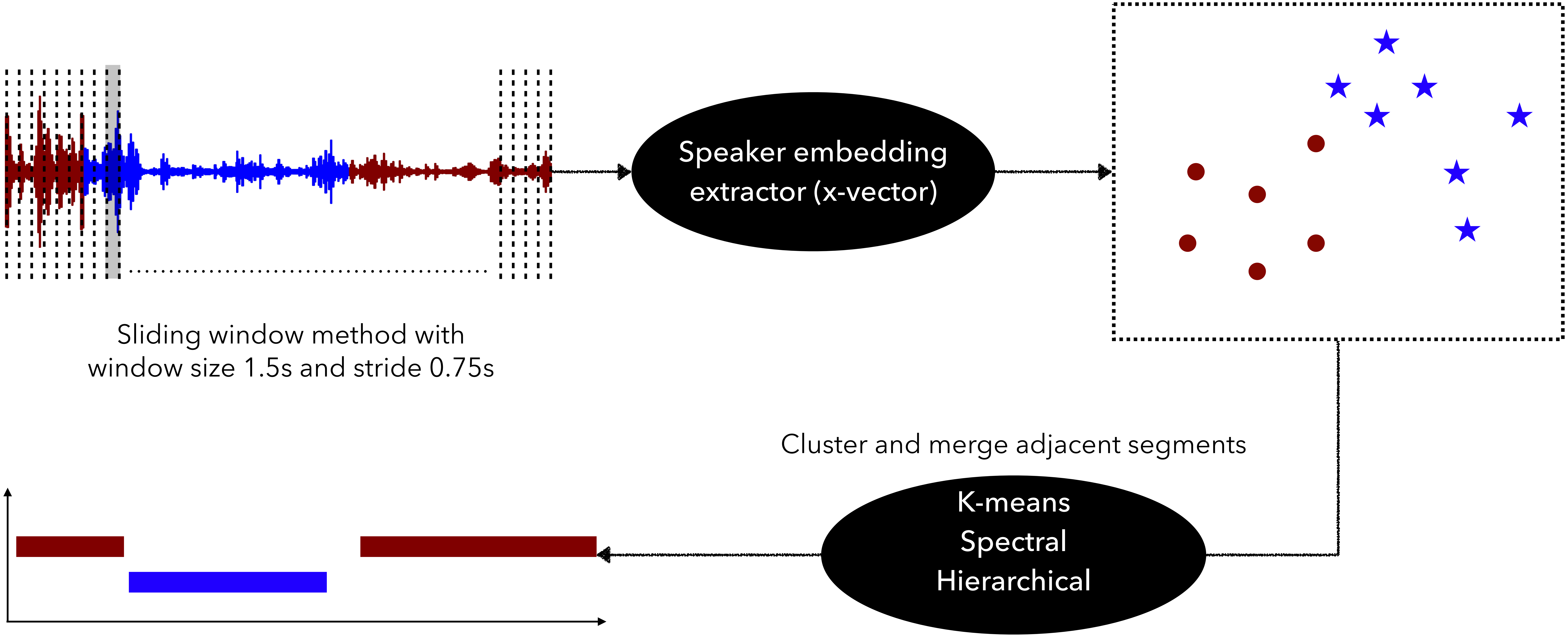
- No need for overlap detection and assignment
- Speaker embedding extractors trained on single-speaker utterances work just fine

- Need to **adapt clustering algorithms** for cross-stream clustering

# Adapting clustering algorithms

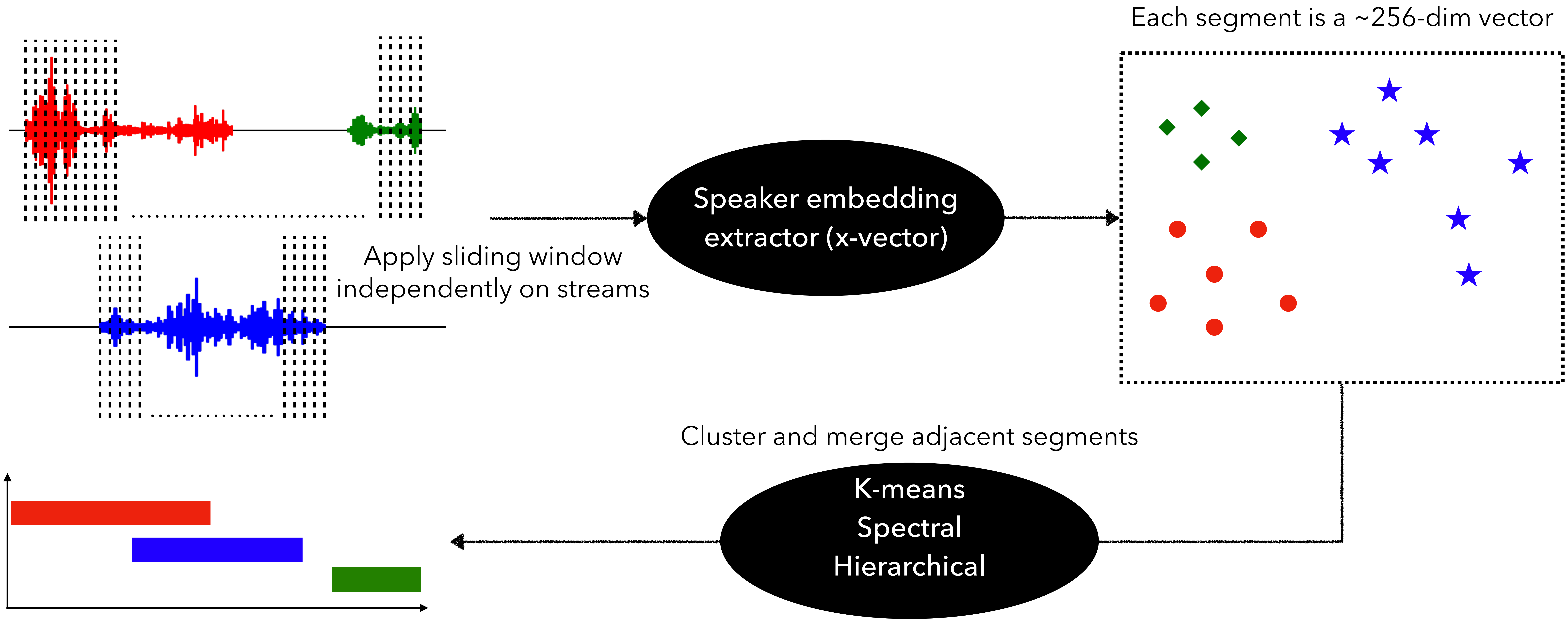
## Sequence-agnostic clustering methods

Each segment is a ~256-dim vector



# Adapting clustering algorithms

## Sequence-agnostic clustering methods



# Adapting clustering algorithms

## Sequence-agnostic clustering methods

- How does it compare with overlap assignment?
- Performance on **LibriCSS**:

| Method | Miss | F.Alarm | Conf. | DER |
|---|---|---|---|---|
| **Spectral + OVL** | 3.8 | 2.2 | 5.3 | 11.3 |
| **CSS + Spectral** | 3.4 | 3.4 | 1.9 | 8.7 |

- Cons: requires a well-trained CSS network

# Adapting clustering algorithms

## Sequence-dependent clustering methods

- These methods perform clustering over the sequential input
- Need special treatment to adapt to the case of separated streams
- Case study: VBx (Bayesian HMM clustering of x-vector sequences)

# The VBx method for diarization

## Preliminary: Variational Bayes

- Observation **X** and latent variable **Z**
- Need to compute **posterior** $p(\mathbf{Z} \mid \mathbf{X})$

$$p(\mathbf{Z} \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \mathbf{Z})p(\mathbf{Z})}{\int p(\mathbf{X} \mid \mathbf{Z})p(\mathbf{Z})\mathrm{d}\mathbf{Z}}$$

- Hard to compute the marginal term in the denominator
- So we will approximate the posterior with some distribution $q(\mathbf{Z})$

# The VBx method for diarization

## Preliminary: Variational Bayes

- Minimize the **KL-divergence**

$$q^*(\mathbf{Z}) = \operatorname{argmin}_{q(\mathbf{Z}) \in Q} \mathrm{KL}(q(\mathbf{Z}) \,||\, p(\mathbf{Z} \mid \mathbf{X}))$$

- Here $Q$ is some family of distributions

- This is equivalent to **maximizing the ELBO**

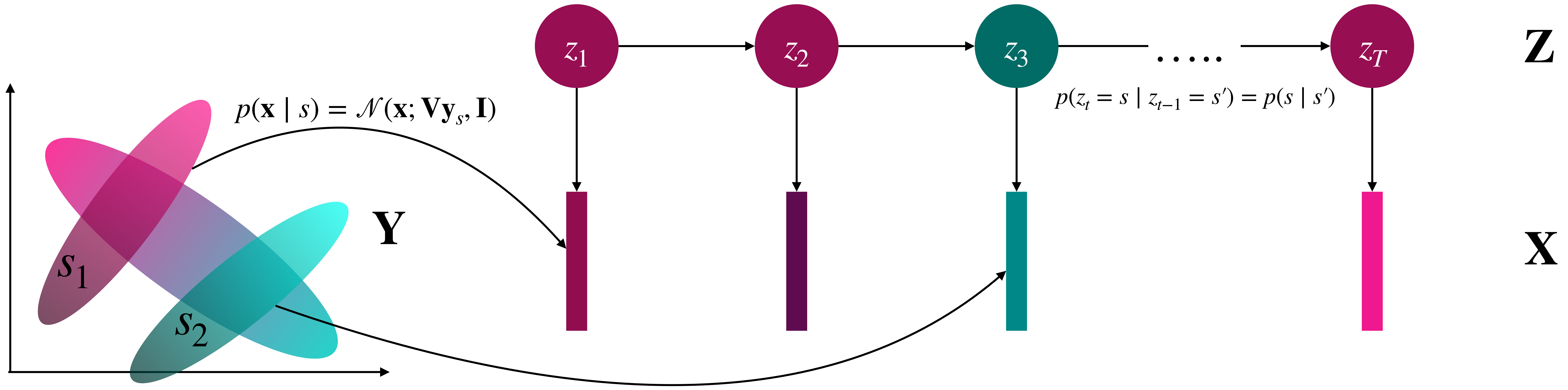
$$\mathrm{ELBO}(q) = \underbrace{\boxed{\mathbb{E}_{q(\mathbf{Z})} \left[\log p(\mathbf{X} \mid \mathbf{Z})\right]}}_{\text{Maximize the likelihood}} - \underbrace{\boxed{\mathrm{KL}(q(\mathbf{Z}) \,||\, p(\mathbf{Z}))}}_{\text{Keep it close to the prior}}$$

- Mean-field approximation: $q(\mathbf{Z}) = \prod_{j=1}^{m} q_j(z_j)$

# The VBx method for diarization

## Setup

- Discrete latent sequence of speakers $\mathbf{Z}$
- Observation: sequence of x-vectors $\mathbf{X}$
- $\mathbf{X}$ is generated from $\mathbf{Z}$ using a *Bayesian Hidden Markov model*



# The VBx method for diarization

## Variational inference

- Computing the **posterior**:

$$p(\mathbf{Z} \mid \mathbf{X}) = \int p(\mathbf{Z}, \mathbf{Y} \mid \mathbf{X}) \mathrm{d}\mathbf{Y}$$

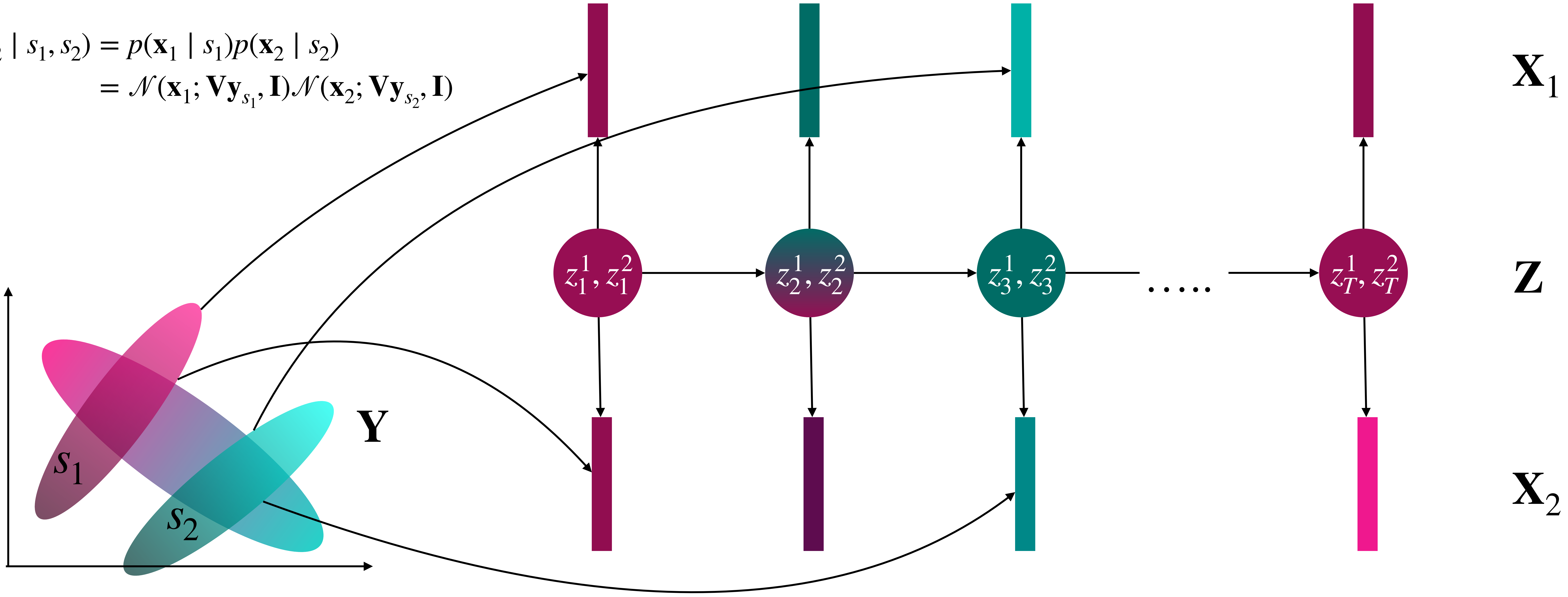
- Again hard because of marginal term in denominator, so use approximation
- Mean-field approximation: $q(\mathbf{Z}, \mathbf{Y}) = q(\mathbf{Z})q(\mathbf{Y})$
- Solved by **maximizing the ELBO**:

$$\mathrm{ELBO}(q) = \mathbb{E}_{q(\mathbf{Z},\mathbf{Y})}\left[\log p(\mathbf{X} \mid \mathbf{Y}, \mathbf{Z})\right] - \mathbb{E}_{q(\mathbf{Y})}\left[\log \frac{q(\mathbf{Y})}{p(\mathbf{Y})}\right] - \mathbb{E}_{q(\mathbf{Z})}\left[\log \frac{q(\mathbf{Z})}{p(\mathbf{Z})}\right]$$

# Extending VBx for CSS output

## Fully-coupled model

$$p(\mathbf{x}_1, \mathbf{x}_2 \mid s_1, s_2) = p(\mathbf{x}_1 \mid s_1)p(\mathbf{x}_2 \mid s_2)$$
$$= \mathcal{N}(\mathbf{x}_1; \mathbf{V}\mathbf{y}_{s_1}, \mathbf{I})\mathcal{N}(\mathbf{x}_2; \mathbf{V}\mathbf{y}_{s_2}, \mathbf{I})$$



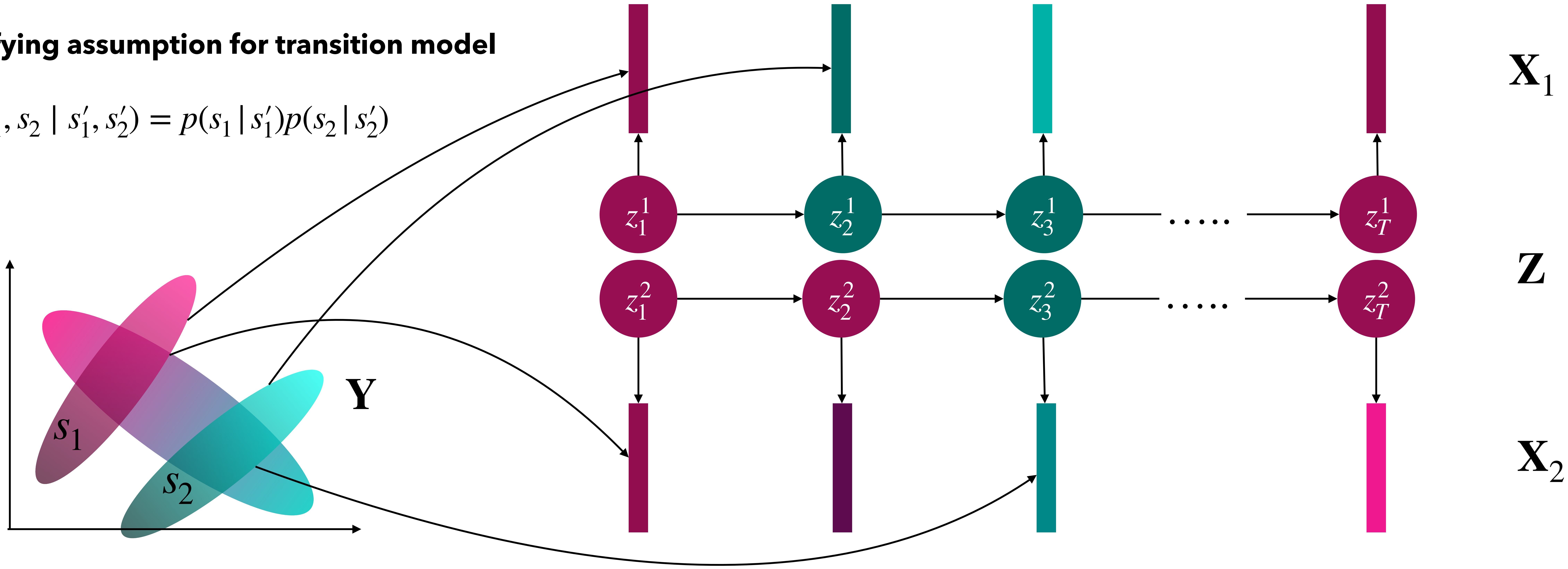

$$p(z_t^1 = s_1, z_t^2 = s_2 \mid z_{t-1} = s_1', z_{t-1}^2 = s_2') = p(s_1, s_2 \mid s_1', s_2')$$

# Extending VBx for CSS output

## State-decoupled model

**Simplifying assumption for transition model**

$$p(s_1, s_2 \mid s'_1, s'_2) = p(s_1 \mid s'_1) p(s_2 \mid s'_2)$$

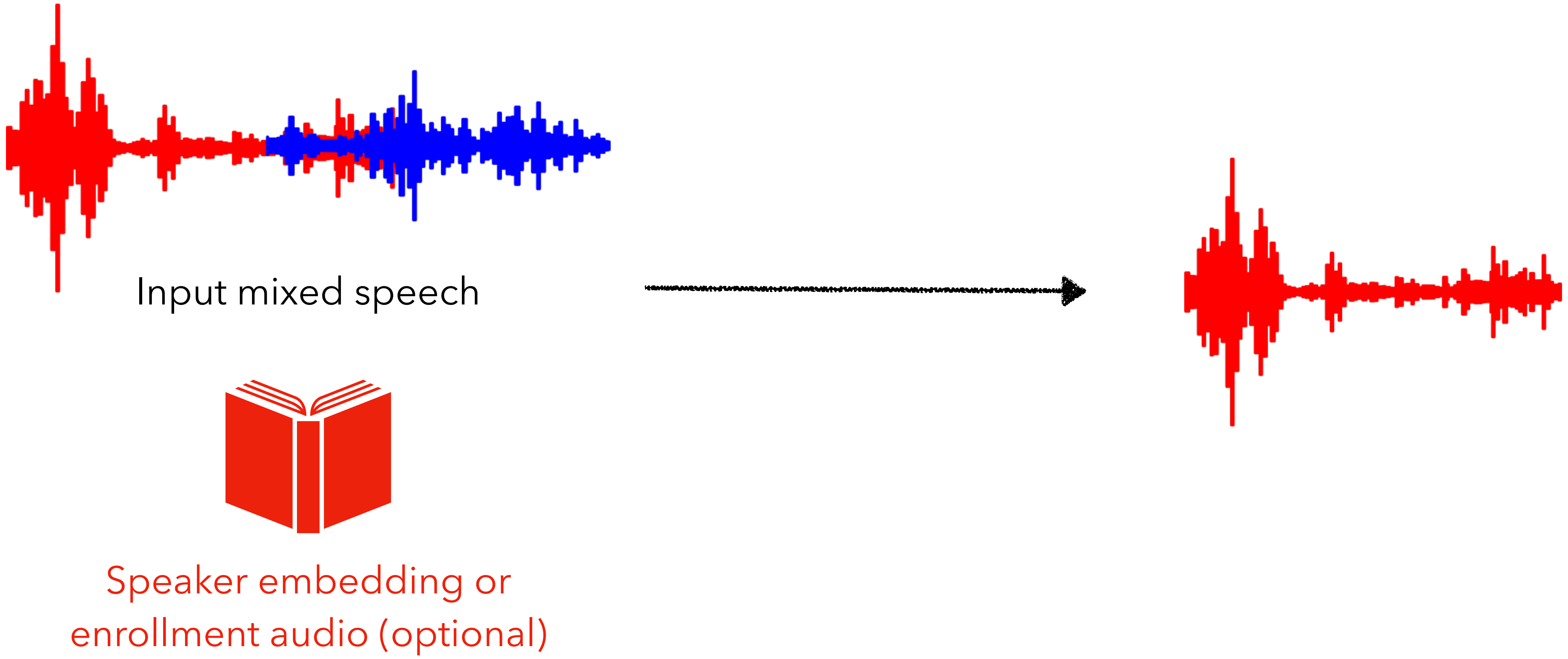


# Target-speaker extraction with GSS

# What is target-speaker extraction?

## Supervised and unsupervised methods

- Given an audio containing mixed speech, extract the speech of a **target speaker**
- Auxiliary information: enrollment audio or speaker embedding



# Guided source separation

## Setup

- Let $\mathbf{Y}_{t,f}$ be a multi-channel input signal in STFT domain, i.e., $\mathbf{Y}_{t,f} \in \mathbb{C}^D$
- We assume the following model of the signal:

Sum of reverb tails

$$\mathbf{Y}_{t,f} = \boxed{\sum_k \mathbf{X}^{\text{early}}_{t,f,k}} + \boxed{\sum_k \mathbf{X}^{\text{tail}}_{t,f,k}} + \boxed{\mathbf{N}_{t,f}}$$

Sum of speaker signals

Noise

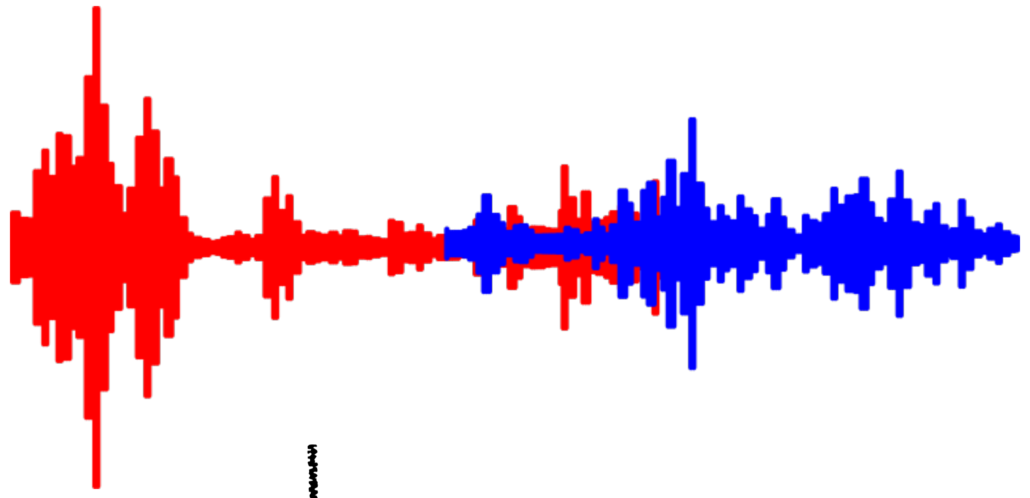
- We want to estimate $\mathbf{X}^{\text{early}}_{t,f,k}$ for a given speaker $k$

# Guided source separation

## Consists of 3 main steps

$$\mathbf{Y}_{t,f} = \underbrace{\sum_k \mathbf{X}^{\text{early}}_{t,f,k}}_{\text{Sum of speaker signals}} + \overbrace{\sum_k \mathbf{X}^{\text{tail}}_{t,f,k}}^{\text{Sum of reverb tails}} + \underbrace{\mathbf{N}_{t,f}}_{\text{Noise}}$$

$\mathbf{Y}_{t,f}$



↓

**De-reverberation using Weighted Prediction Error (WPE)** — Remove the late reverb

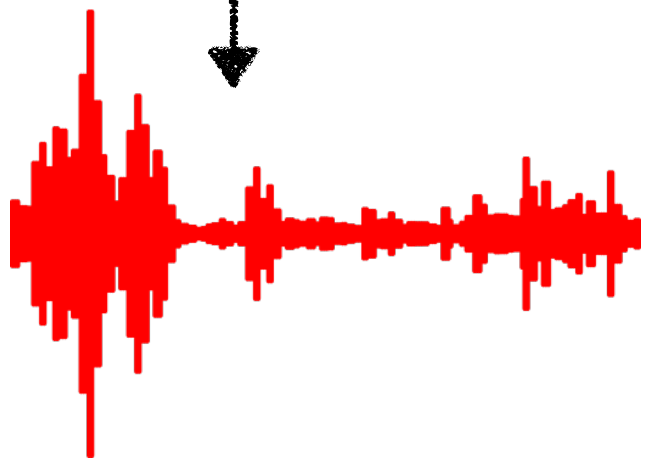
↓

**Mask estimation using complex angular central GMM (CACGMM)** — Estimate T-F masks for all speakers and noise

↓

**Mask-based MVDR beamforming** — Use T-F masks to extract desired signal from input

↓



# Guided source separation

## Step 1: De-reverberation using WPE

$$\mathbf{Y}_{t,f}^{\text{early}} = \mathbf{Y}_{t,f}^{1} - \boxed{\hat{\mathbf{g}}_f^{H}\mathbf{Y}_{t-\tau,f}}$$

Multi-channel linear regression

- Assume $\mathbf{Y}_{t,f}^{\text{early}}$ for each T-F bin is modeled by a zero-mean complex Gaussian with variance $\lambda_{t,f}$
- Parameters to estimate: $\lambda_{t,f}$ for every time-frequency and $\mathbf{g}_k$ for every frequency
- Use maximum likelihood estimation (iteratively solve for parameters)

# Guided source separation

## Step 2: Mask estimation using CACGMMs

- Assume $\tilde{\mathbf{Y}}_{t,f}$ for each T-F bin is modeled as a mixture of complex angular central Gaussians

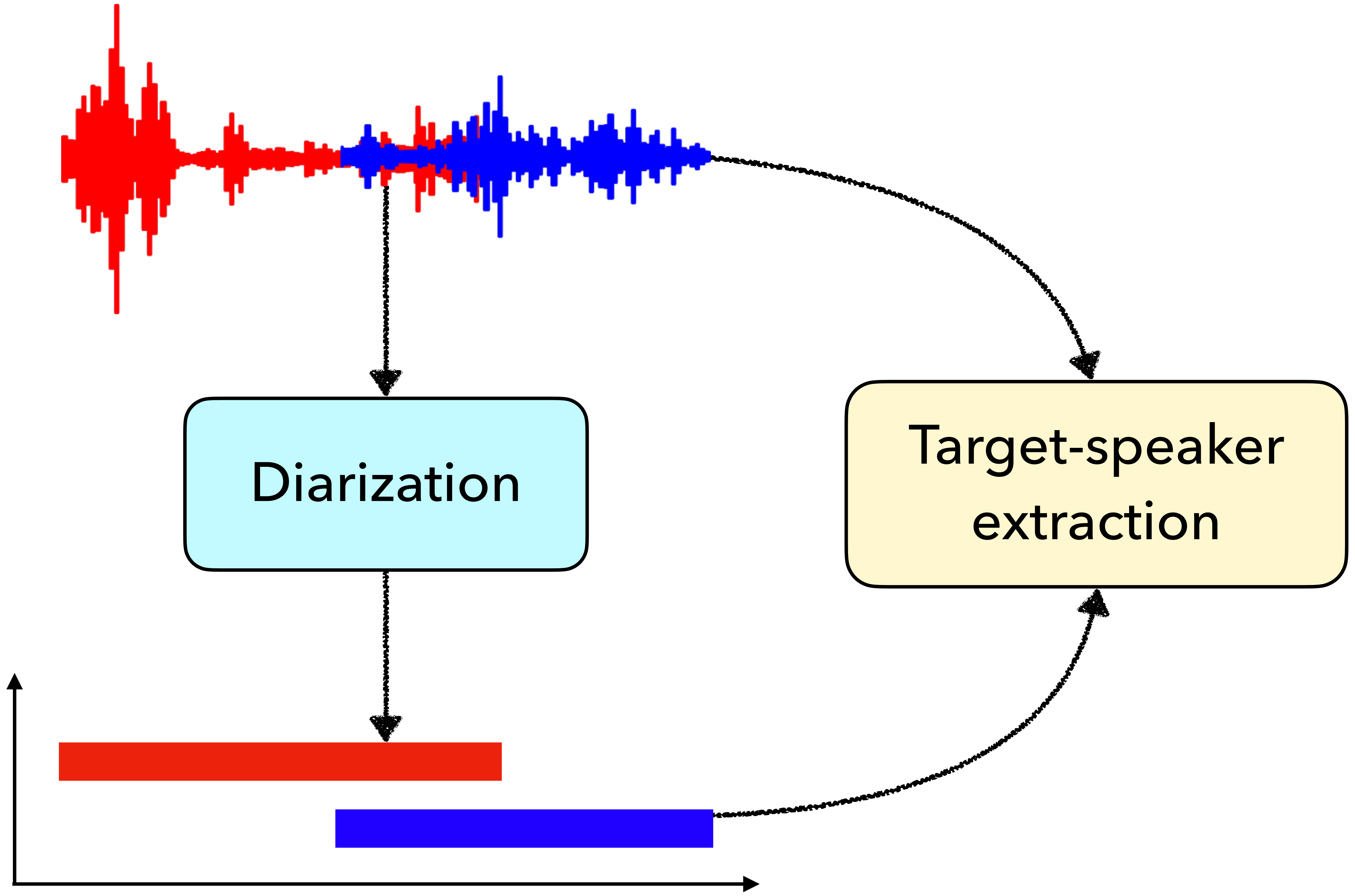
$$p(\tilde{\mathbf{Y}}_{t,f}) = \sum_k \pi_{f,k} \mathscr{A}(\tilde{\mathbf{Y}}_{t,f}; \mathbf{B}_{f,k})$$

- Here, $\mathbf{B}_{f,k}$ is a positive-definite Hermitian matrix that controls the CACG
- Cannot directly run EM algorithm:

    1. Need to know number of mixture components $k$
    2. Permutation problem for speaker indices for different $f$

# Guided source separation

## Step 2: Mask estimation using CACGMMs




- Use diarization output!
- Number of components = number of speakers + 1 (for noise)
- Fix the global speaker order according to diarization output

# Guided source separation

## Step 2: Mask estimation using CACGMMs

- Apply E-M algorithm
- **E-step:** Compute state posteriors at each time-step

$$\gamma_{t,f,k} = \frac{\pi_{t,f,k} |\mathbf{B}_{f,k}|^{-1} (\tilde{\mathbf{Y}}_{t,f}^H \mathbf{B}^{-1} \tilde{\mathbf{Y}}_{t,f})^{-D}}{\sum_{k'} \pi_{t,f,k'} |\mathbf{B}_{f,k'}|^{-1} (\tilde{\mathbf{Y}}_{t,f}^H \mathbf{B}^{-1} \tilde{\mathbf{Y}}_{t,f})^{-D}}$$

- **M-step:** Compute mixture weights and covariance
- Finally, $\gamma_{t,f,k}$ gives the **T-F masks** of all the speakers and noise

# Guided source separation

## Step 3: Mask-based MVDR beamforming

- Signal consists of a combination of target and distortion

$$\mathbf{Y}_{t,f} = \mathbf{d}_f \mathbf{S}_{t,f} + \mathbf{N}_{t,f}$$

- Here, $\mathbf{d}$ is called the steering vector
- A beamformer tries to **weight the sum of multi-channel signal** into enhanced signal

$$\hat{\mathbf{S}} = \mathbf{w}^H \mathbf{Y}, \quad \mathbf{w} \in \mathbb{C}^{D \times F}$$

- If weight of frequency bin is constant for all time steps, called time-invariant

# Guided source separation

## Step 3: Mask-based MVDR beamforming

- MVDR beamformer: minimum variance distortionless response
- Minimize the power of the interfering signal while preserving the distortionless source signal

$$\mathbf{w}_{\mathrm{MVDR}}(f) = \arg\min_{\mathbf{w}} \mathbf{w}^{\mathrm{H}}(f)\Phi_{\mathbf{YY}}(f)\mathbf{w}(f)$$

$$\text{s.t.} \quad \mathbf{w}(f)^{\mathrm{H}}\mathbf{d}(f) = 1$$

- Here, $\Phi_{YY}(f)$ is the covariance of the noisy STFT at frequency $f$.

# Guided source separation

## Step 3: Mask-based MVDR beamforming

$$\gamma_{t,f,k}$$

Target mask

$$\mathbf{\Phi}_k(f) = \frac{1}{T}\sum_t \gamma_{t,f,k}\tilde{\mathbf{Y}}_{t,f}\tilde{\mathbf{Y}}_{t,f}^H$$

Spatial covariance matrix for target

$$\gamma_{t,f,n} = \sum_{k' \neq k} \gamma_{t,f,k'}$$

Distortion mask

$$\mathbf{\Phi}_n(f) = \frac{1}{T}\sum_t \gamma_{t,f,n}\tilde{\mathbf{Y}}_{t,f}\tilde{\mathbf{Y}}_{t,f}^H$$

Spatial covariance matrix for noise

**Beamforming vector**: gives the weight assigned to each channel for every frequency

$$\mathbf{w}_k(f) = \frac{\mathbf{\Phi}_n^{-1}(f)\mathbf{\Phi}_k(f)\mathbf{e}_{\text{ref}}}{\text{tr}\left(\mathbf{\Phi}_n^{-1}(f)\mathbf{\Phi}_k(f)\right)}$$

$$\hat{\mathbf{X}}_{t,f,k} = \tilde{\mathbf{Y}}_{t,f} \cdot \mathbf{w}_k(f)$$

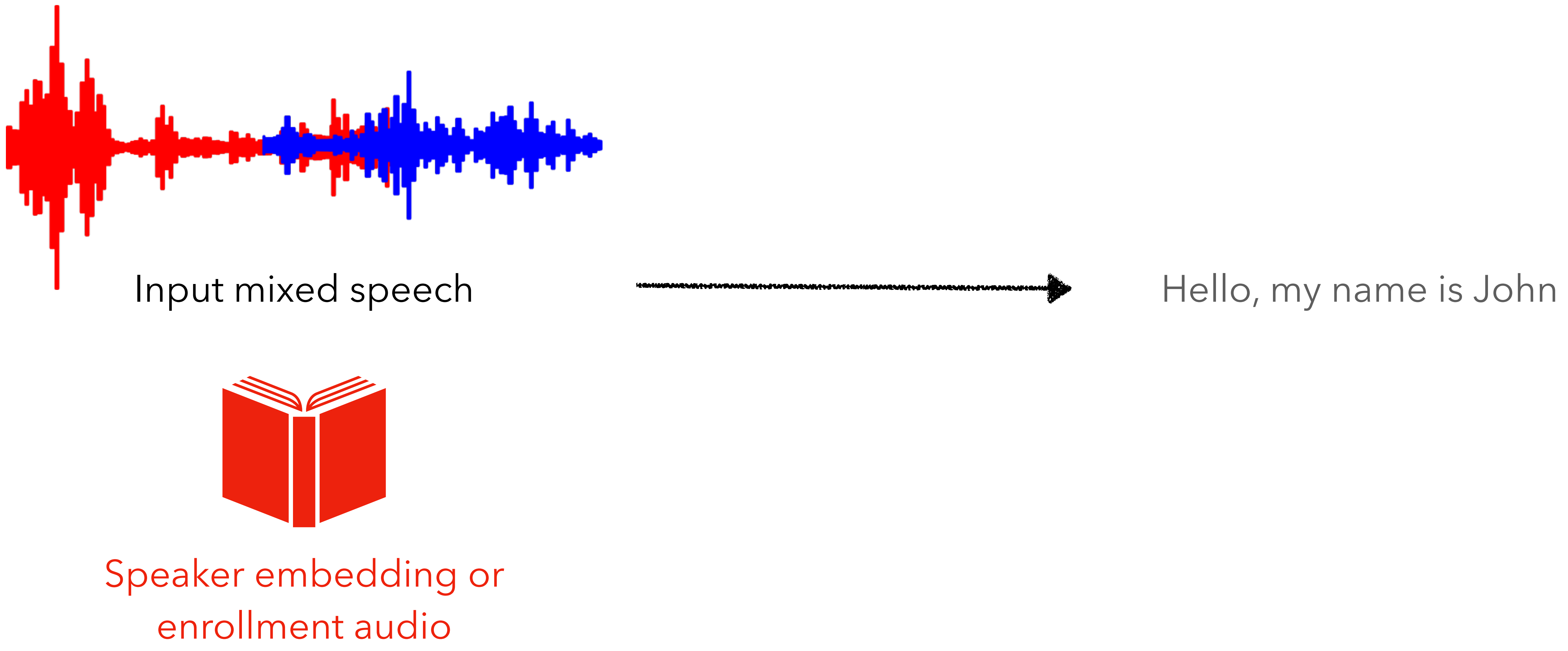
Estimated signal for target $k$

# Target-speaker ASR

# What is target-speaker ASR?

## Target-speaker ASR = target-speaker extraction + ASR

- Given an audio containing mixed speech, transcribe the utterances spoken by a **target speaker**
- Auxiliary information: enrollment audio or speaker embedding



# What is target-speaker ASR?

## Two popular models

- Two popular methods for target-speaker ASR (similar idea)
    1. SpeakerBeam (NTT, Japan)
    2. VoiceFilter (Google)

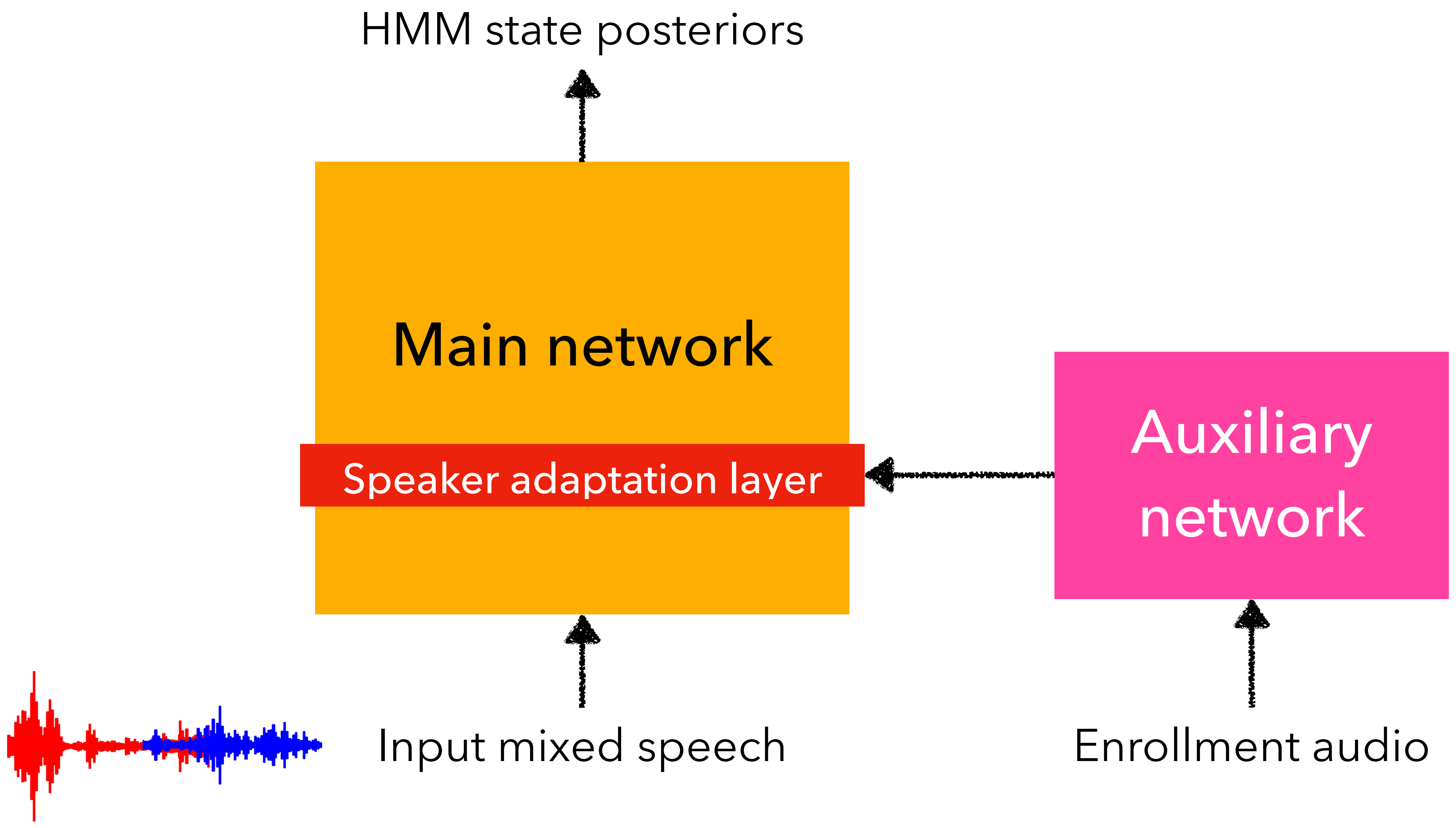
Delcroix, M., Žmolíková, K., Kinoshita, K., Ogawa, A., & Nakatani, T. (2018). Single Channel Target Speaker Extraction and Recognition with Speaker Beam. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5554-5558.

Wang, Q., Lopez-Moreno, I., Saglam, M., Wilson, K.W., Chiao, A., Liu, R., He, Y., Li, W., Pelecanos, J.W., Nika, M., & Gruenstein, A. (2020). VoiceFilter-Lite: Streaming Targeted Voice Separation for On-Device Speech Recognition. *ArXiv, abs/2009.04323*.

# Target-speaker ASR

## SpeakerBeam

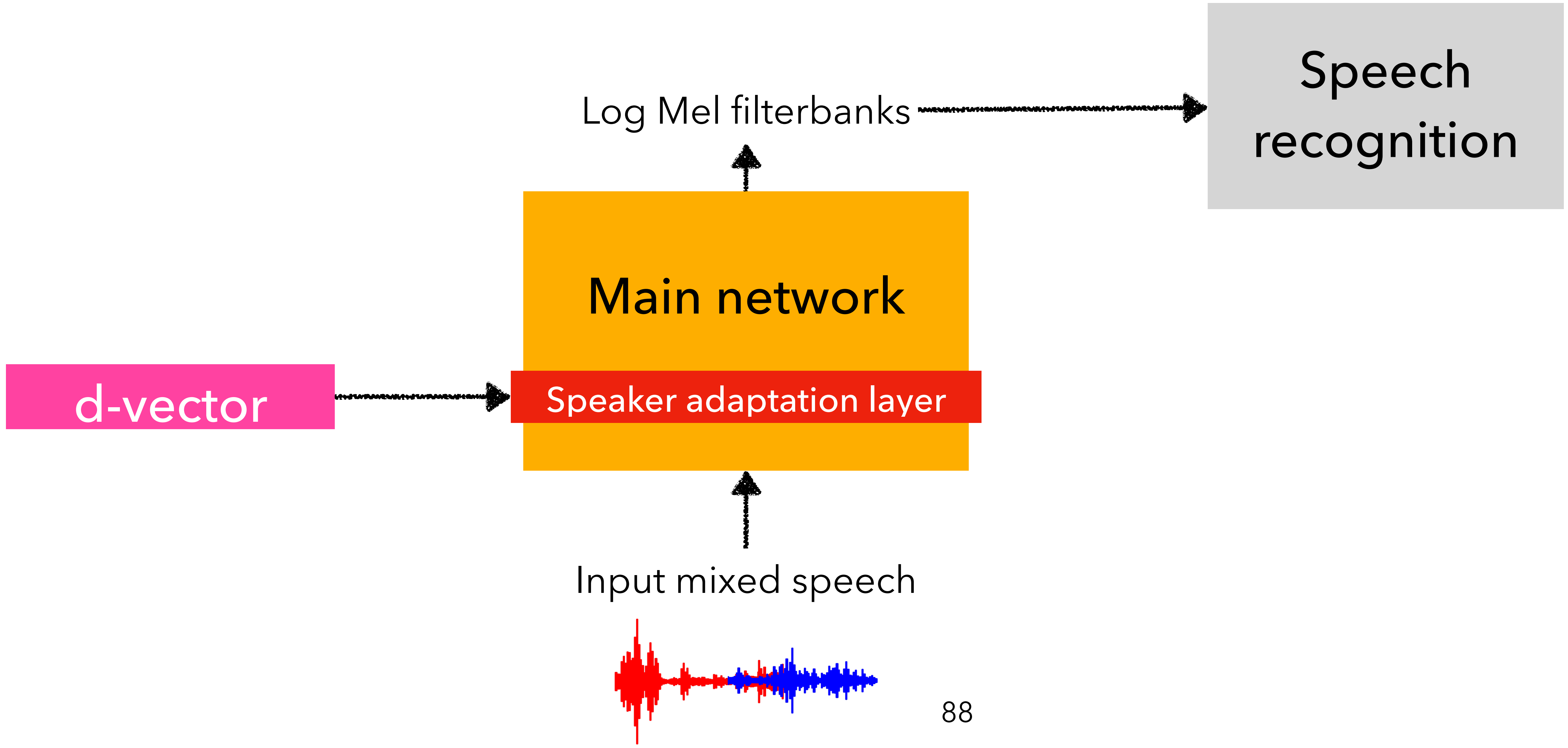
- Use an “auxiliary network” that is trained jointly with the main network, which provides speaker adaptation weights.



# Target-speaker ASR

## VoiceFilter

- Use pre-trained speaker embeddings as auxiliary information (instead of enrollment audio)
- Predict log Mel filterbanks instead of HMM state posteriors
- Techniques to avoid over-suppression



# Target-speaker ASR

## VoiceFilter: avoiding over-suppression

- Voice filtering can cause false deletions when non-speech noise is present; known as "over-suppression"
- Use **asymmetric L2 loss**: penalize more if over-suppressed

$$L_{\text{asym}} = \sum_t \sum_f \left( g_{\text{asym}} \left( S_{\text{cln}}(t,f) - S_{\text{enh}}(t,f), \alpha \right) \right)^2 \qquad g_{\text{asym}}(x, \alpha) = \begin{cases} x & \text{if } x \leqslant 0 \\ \alpha \cdot x & \text{if } x > 0 \end{cases}$$

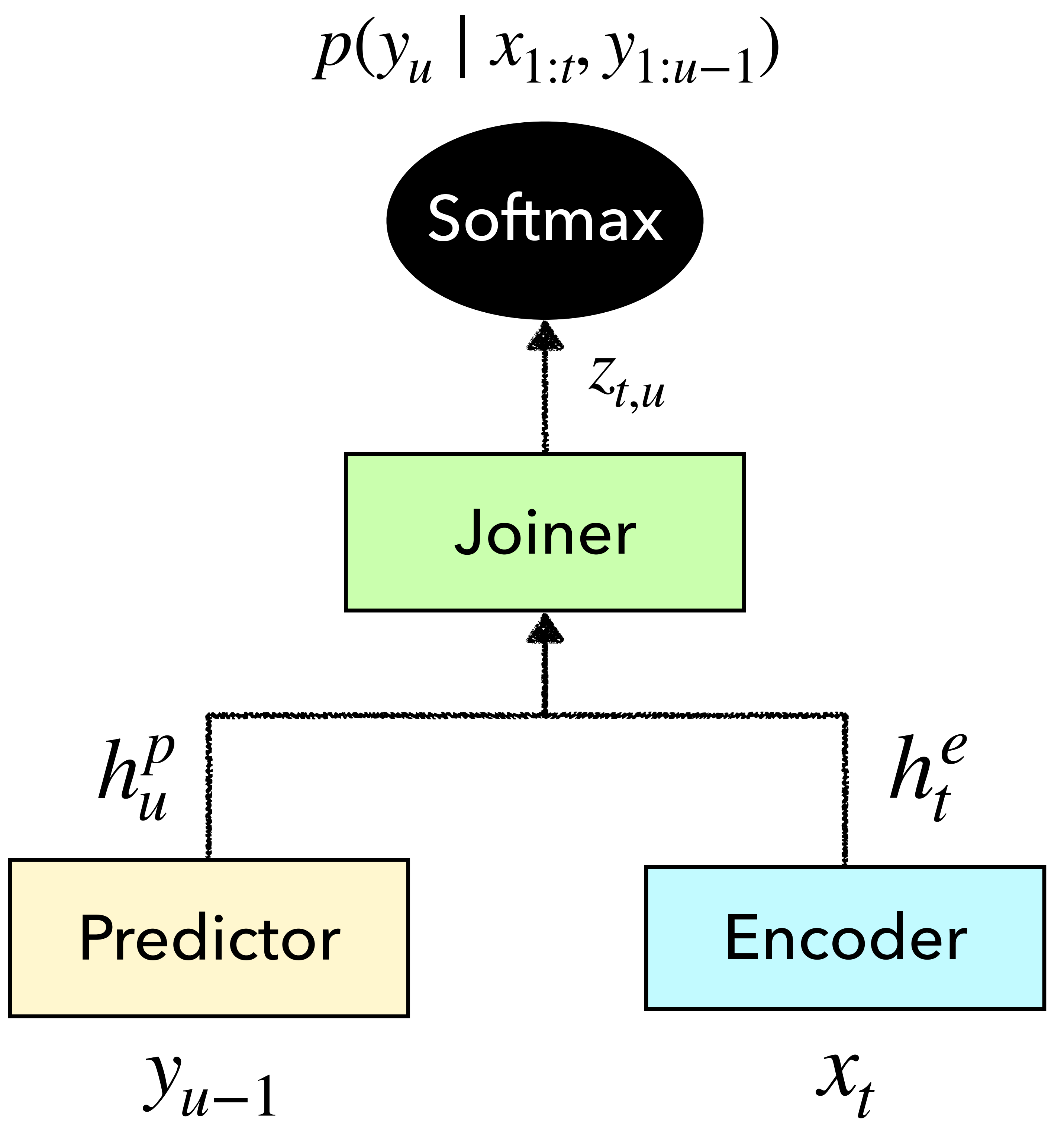
- Use **adaptive suppression strength**:

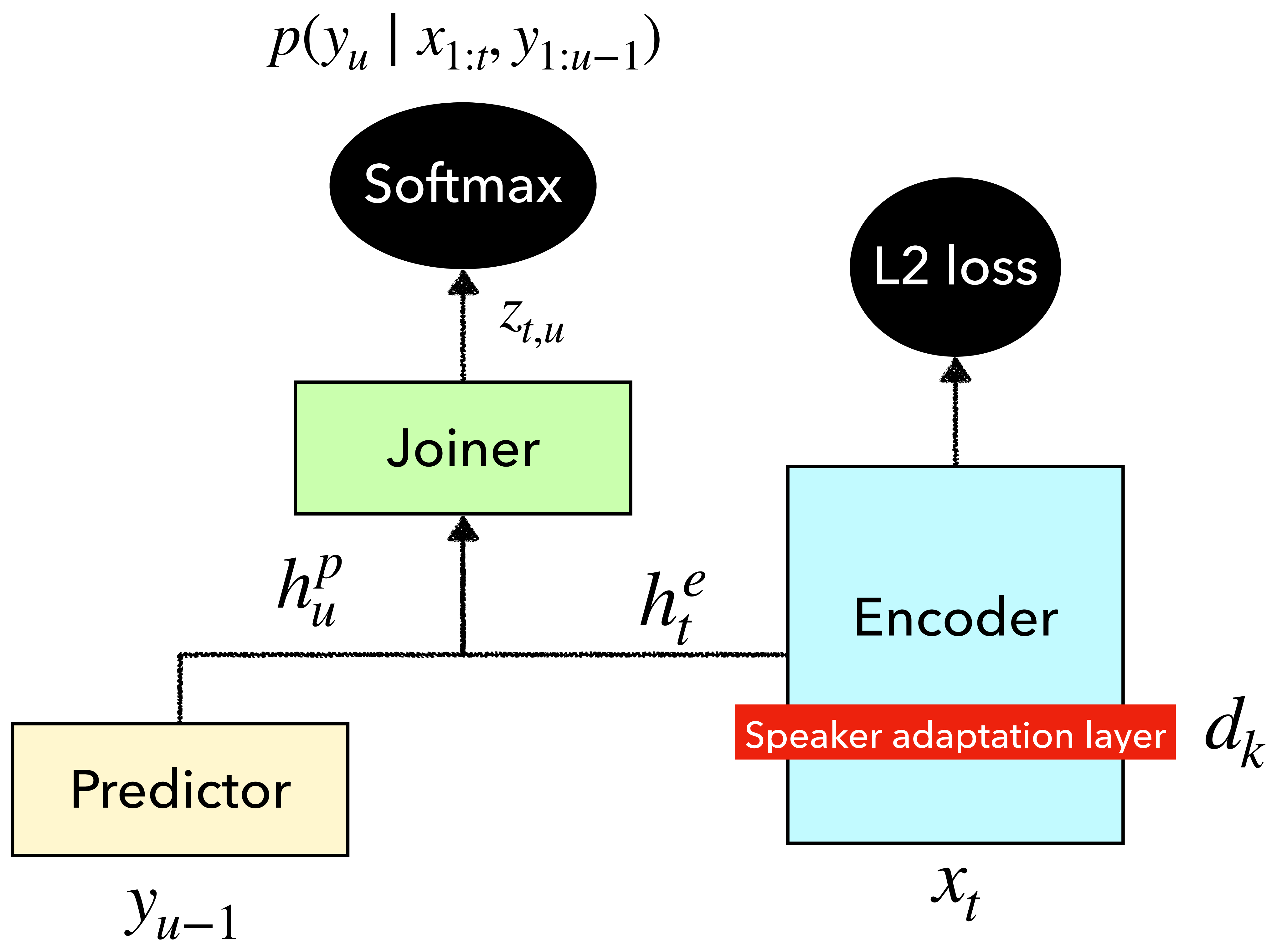
$$S_{\text{out}}^{(t)} = w \cdot S_{\text{enh}}^{(t)} + (1 - w) \cdot S_{\text{in}}^{(t)}$$

# Proposed approach

## TS-ASR based on transducers

- Most industry-grade ASR is built on top of the transducer model
- Use this as the base model and integrate speaker adaptive layer

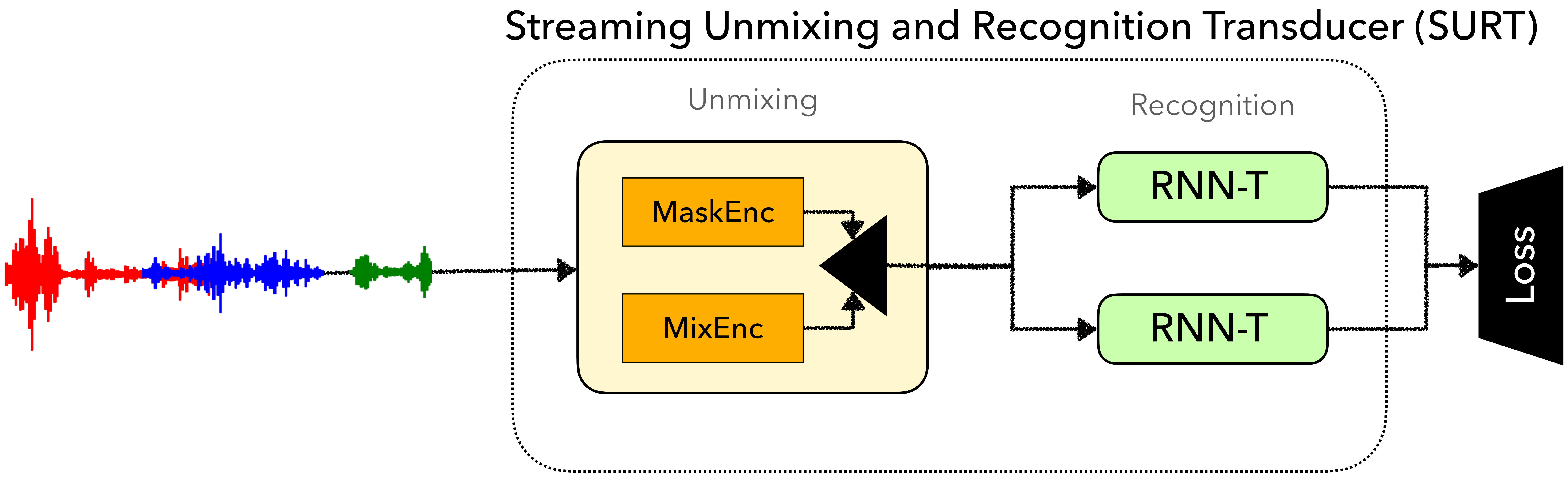


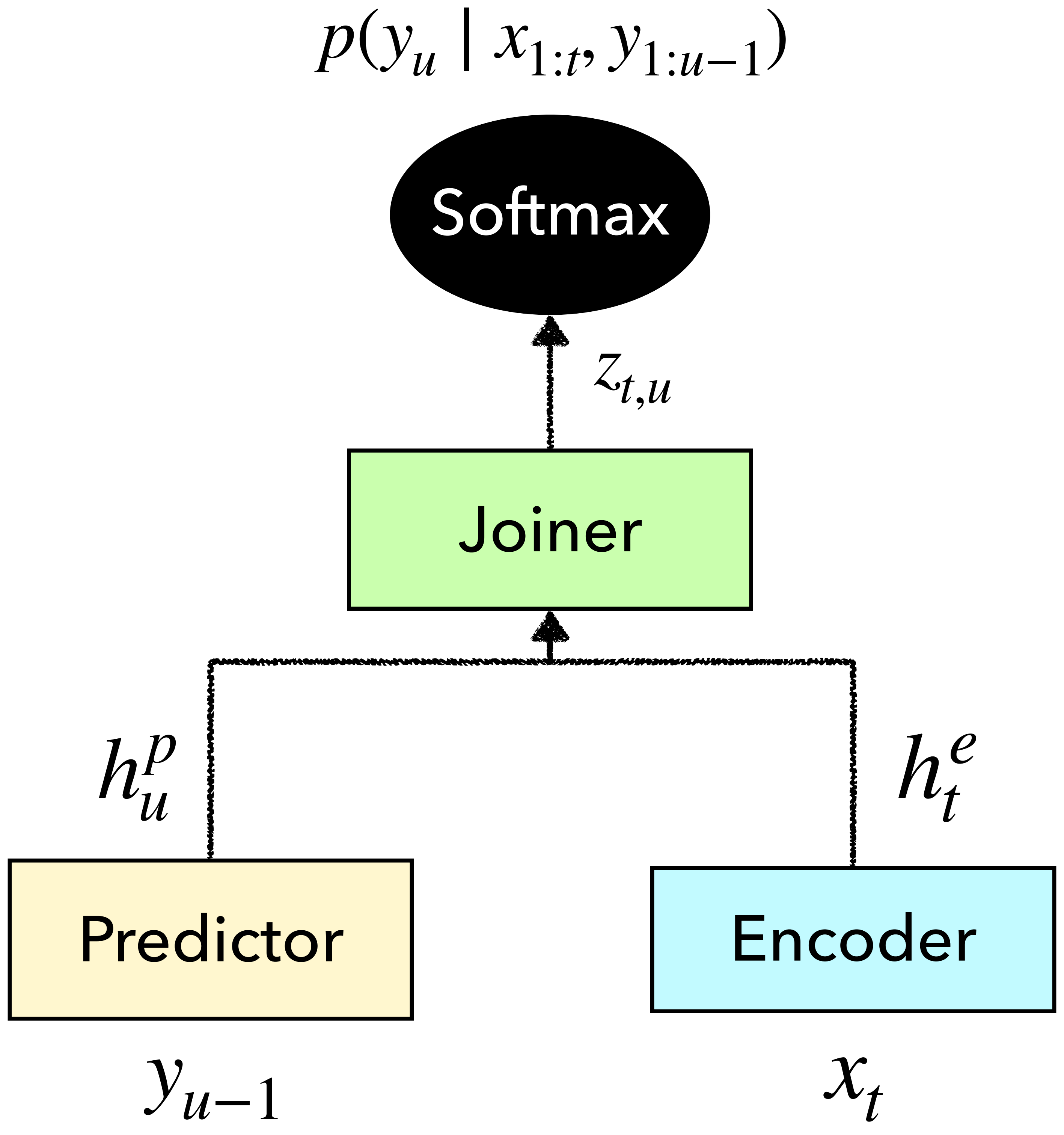



# SURT for long recordings

# Streaming Unmixing and Recognition Transducer

## Basics



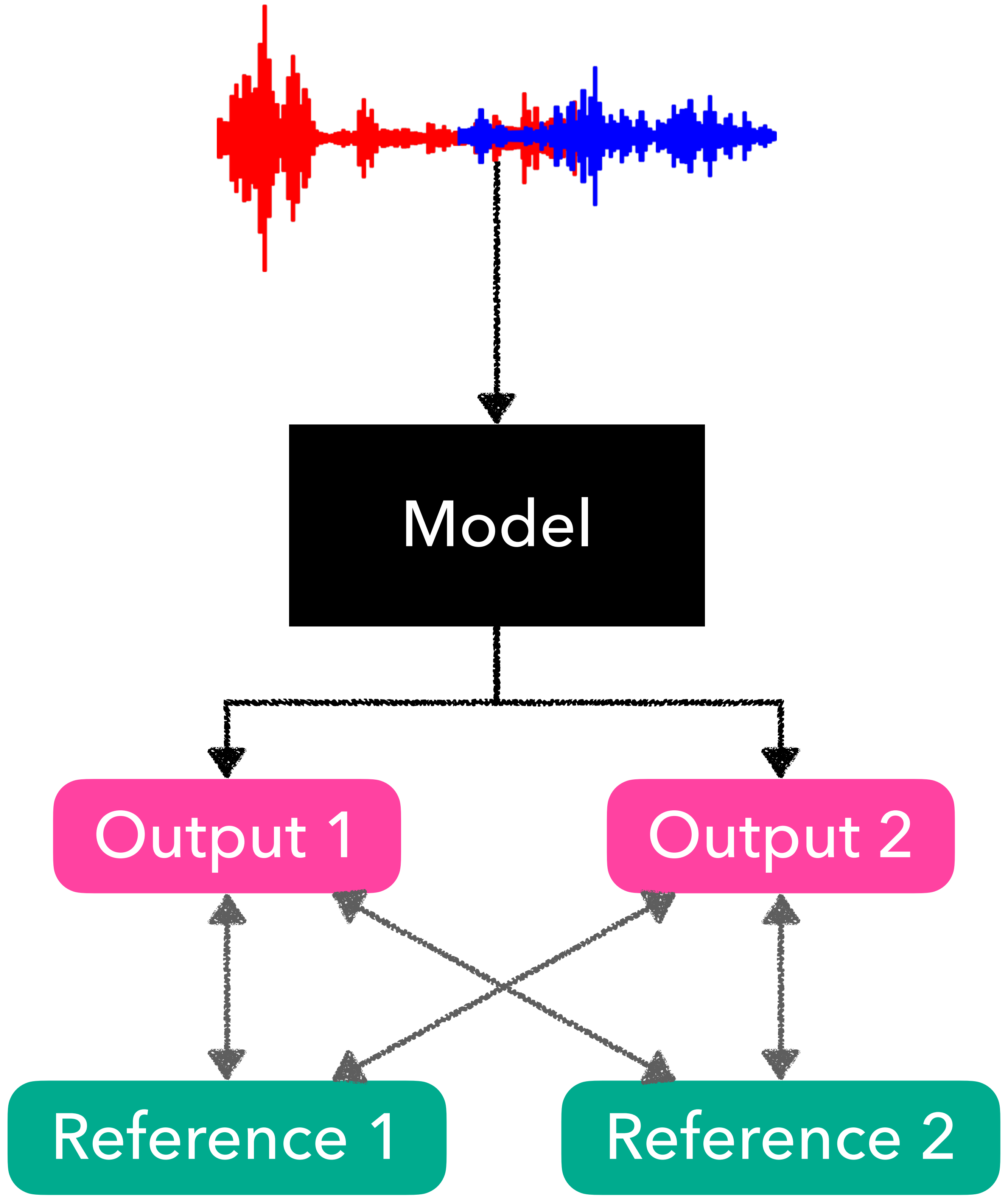




- Unmixing part separates mixed speech into non-overlapping features
- Made of convolutional layers

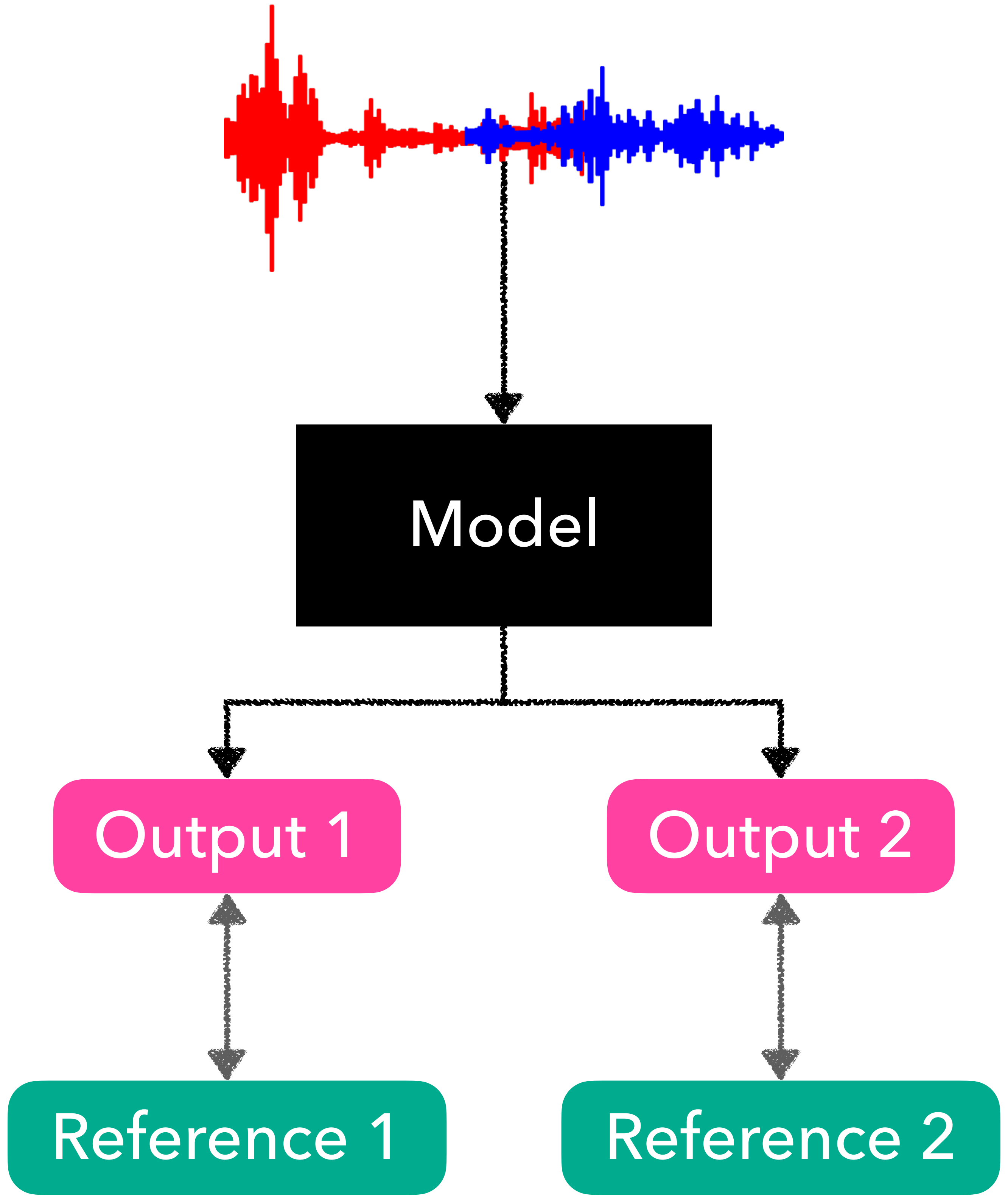
- Transducer model is used as the recognizer
- Output of unmixing component is fed into encoder
- Use HEAT loss over the transducer loss

# Streaming Unmixing and Recognition Transducer

## PIT versus HEAT




Permutation invariant training (PIT)

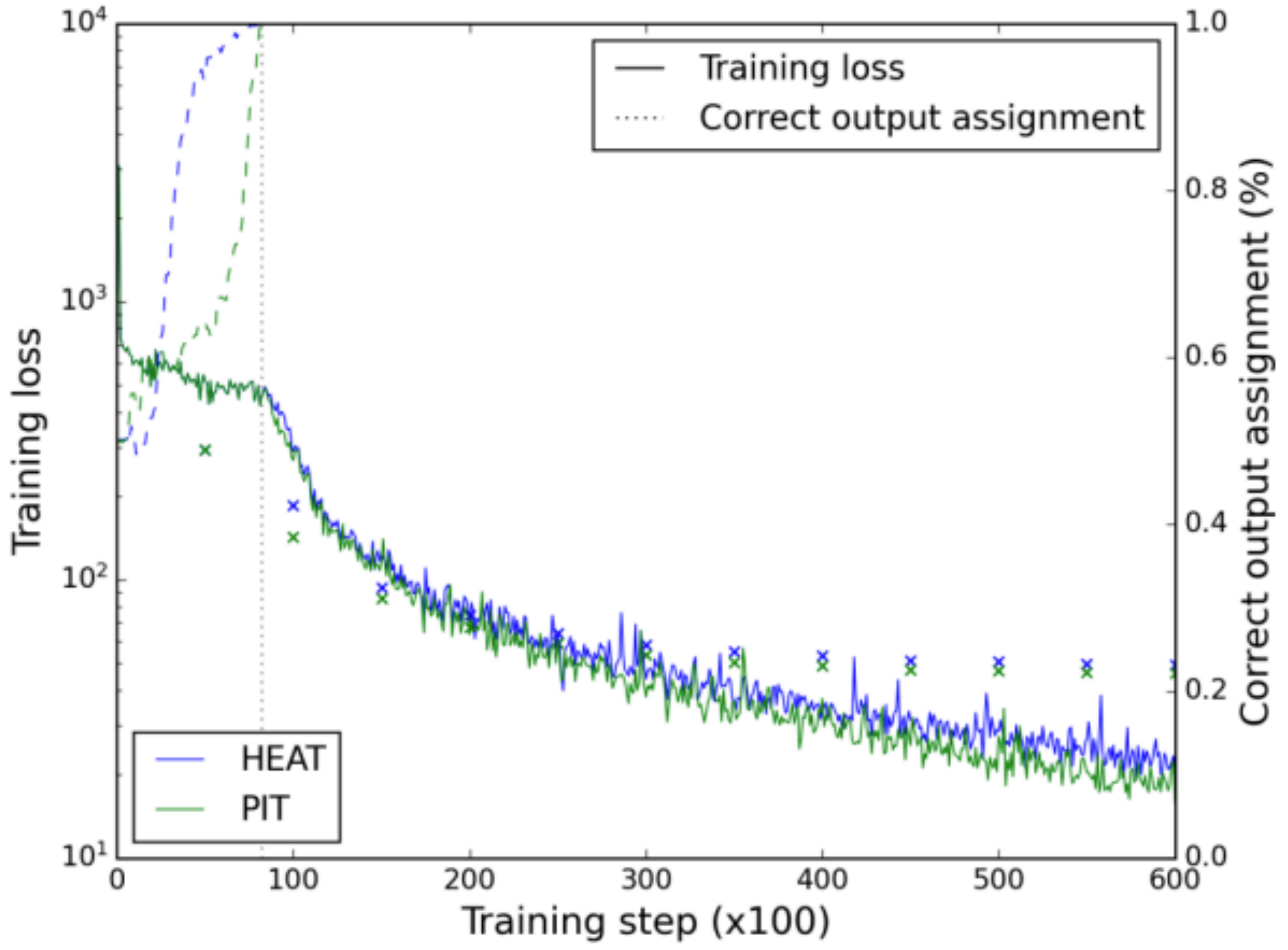



Heuristic error assignment training (HEAT)

# Streaming Unmixing and Recognition Transducer

## PIT versus HEAT

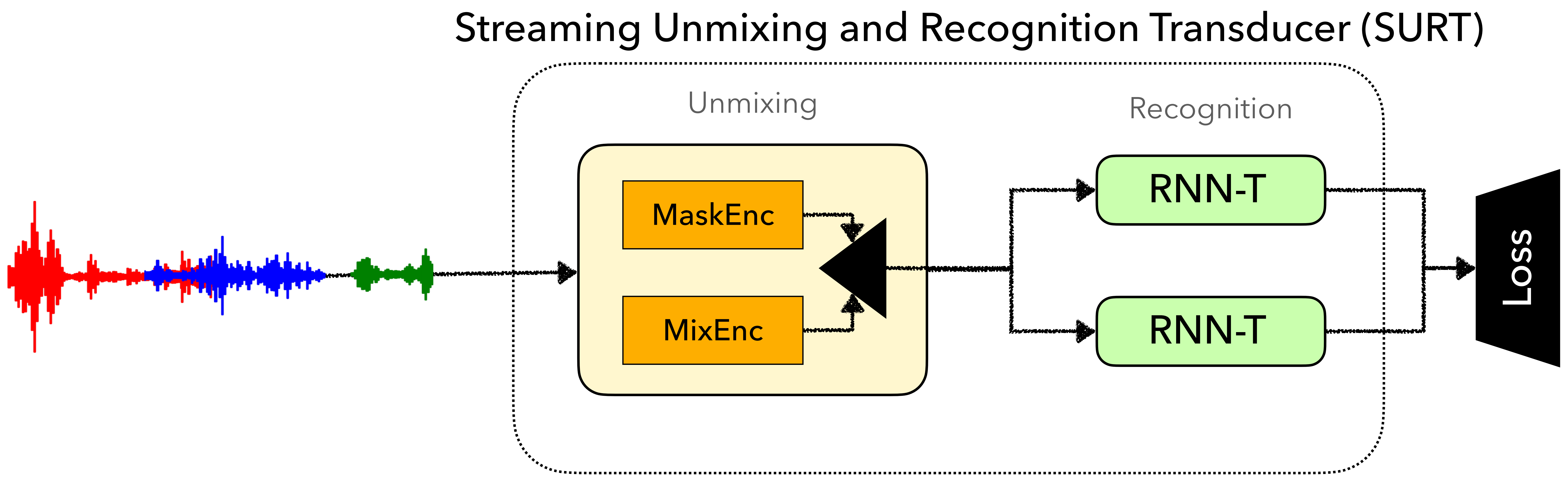
| Permutation invariant training (PIT) | Heuristic error assignment training (HEAT) |
|---|---|
| ✗ Requires computing all permutations of outputs and references | Requires computing only 1 permutation of output and reference ✓ |
| ✗ Can be prohibitively slow when N >> 2 (exponential in N) | Complexity increases linearly with N ✓ |




For utterances with non-zero delay, PIT learns the same heuristic as HEAT

# Streaming Unmixing and Recognition Transducer

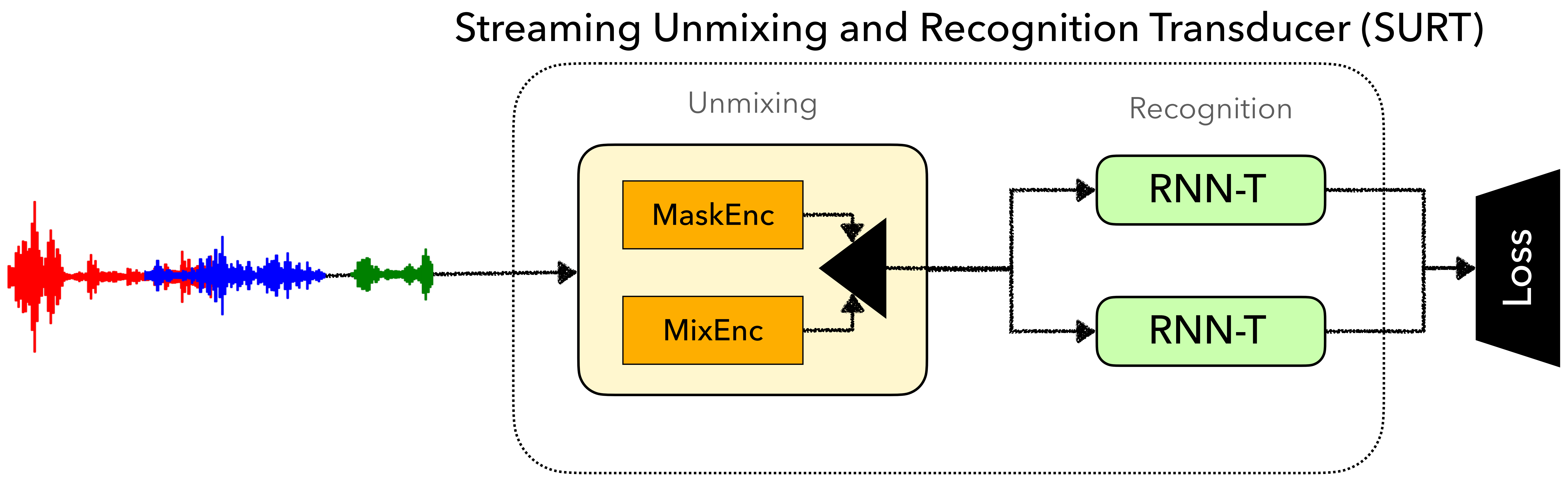
## Problem with vanilla SURT




Vanilla SURT with LSTM-based transducers is not suitable for decoding long recordings

# Streaming Unmixing and Recognition Transducer

## Main changes to make it work

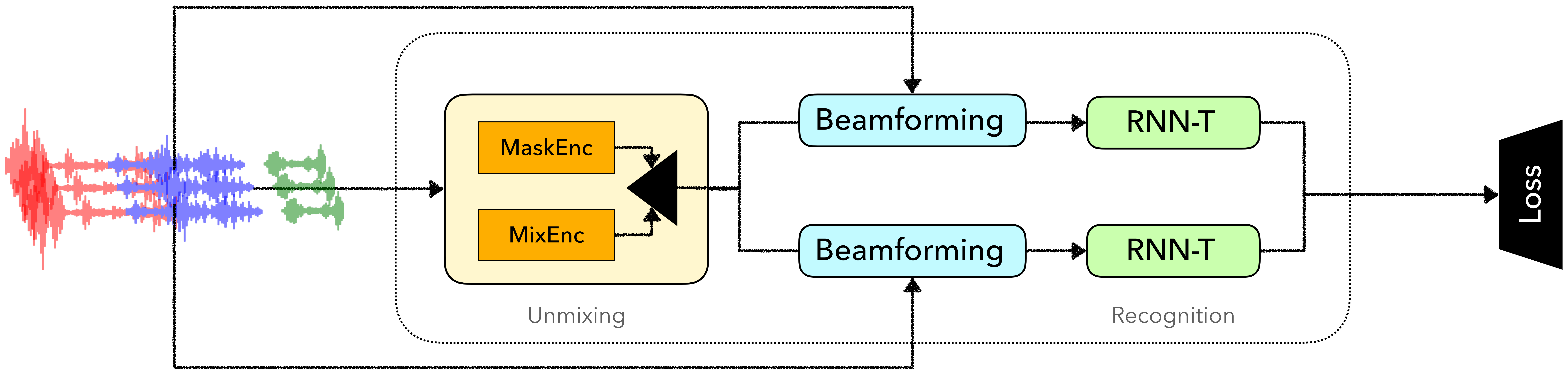



- Use **multi-turn** training data
- Use curriculum learning

- Use **dual-path modeling**, i.e., DP-LSTM and DP-Transformer
- Use chunk width randomization for dual-path model training

# Proposed advances

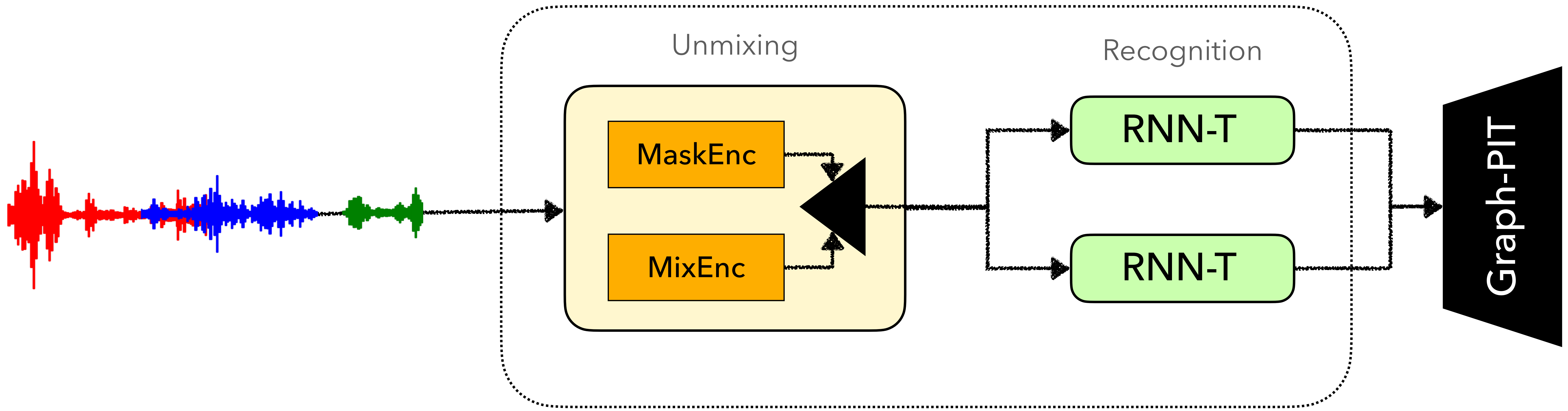
## Multi-channel input




- Use multi-channel input with estimated masks
- **Neural MVDR beamforming**

# Proposed advances

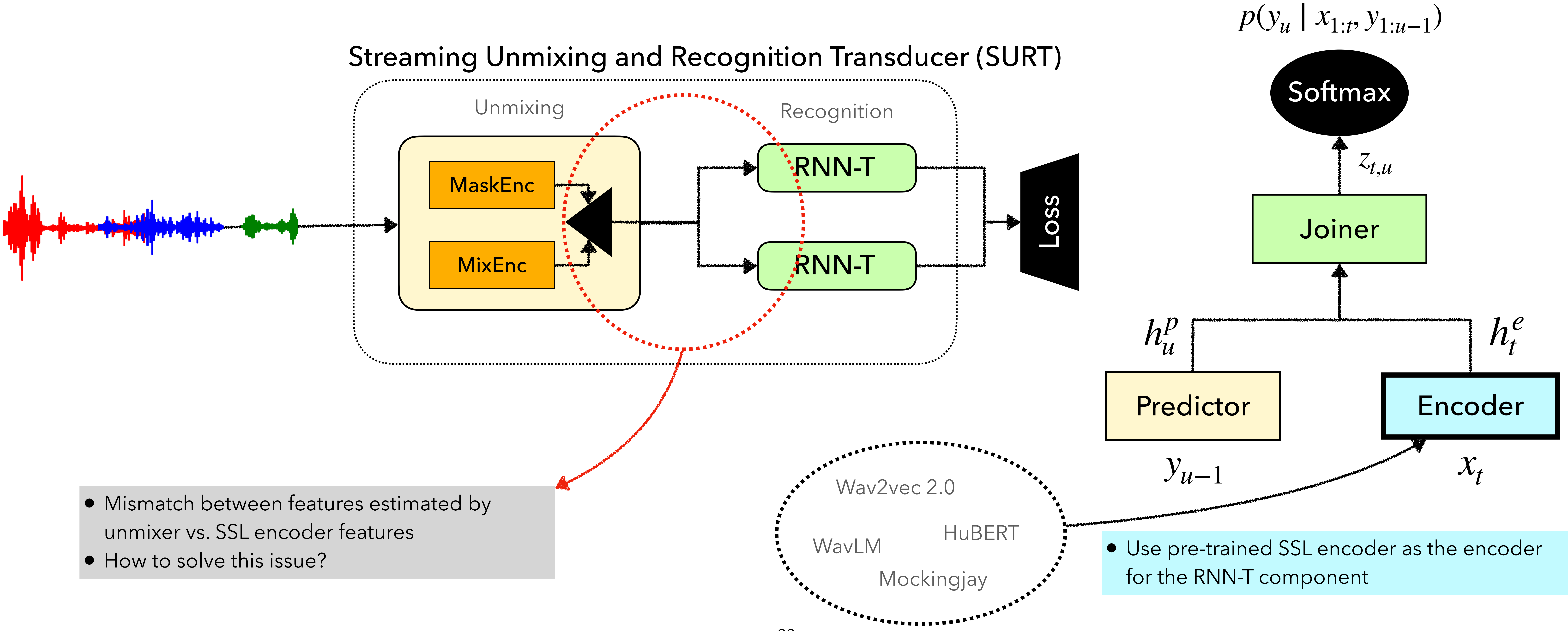
## Training with graph-PIT




- Use Graph-PIT for training instead of HEAT loss
- Provides more flexibility to the model (since we use several possible output assignments)

# Proposed advances

## Self-supervised learning




- Mismatch between features estimated by unmixer vs. SSL encoder features
- How to solve this issue?

- Use pre-trained SSL encoder as the encoder for the RNN-T component

# Fast and efficient SURT

## Integration with k2 and icefall

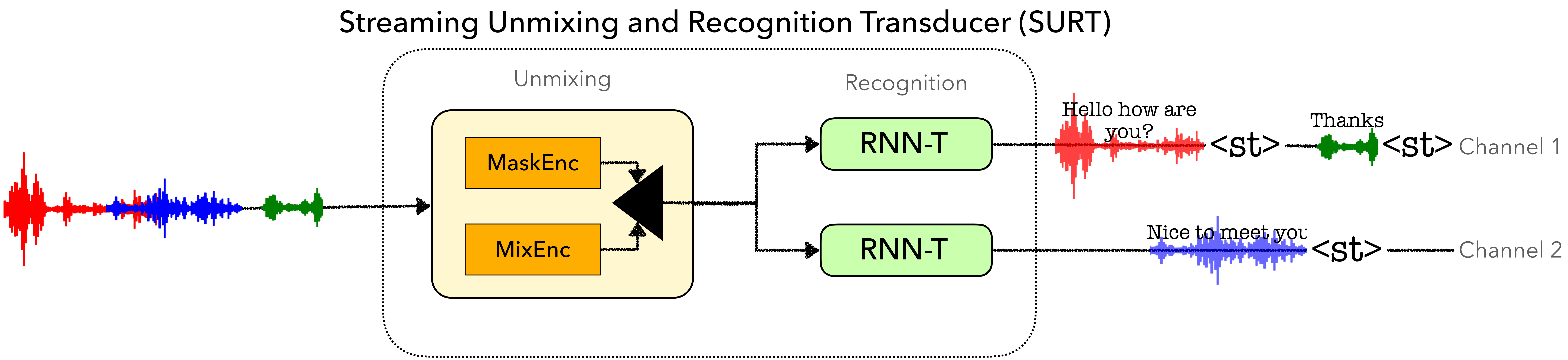
- Monotonic RNN-T topology: emit at most 1 label per time step
- Stateless decoder: replace LSTM with Conv1D
- Pruned joint network to avoid OOM

- **Allows fast decoding and lattice generation with WFST**

https://github.com/k2-fsa/k2
https://github.com/k2-fsa/icefall

# How to perform diarization with SURT?

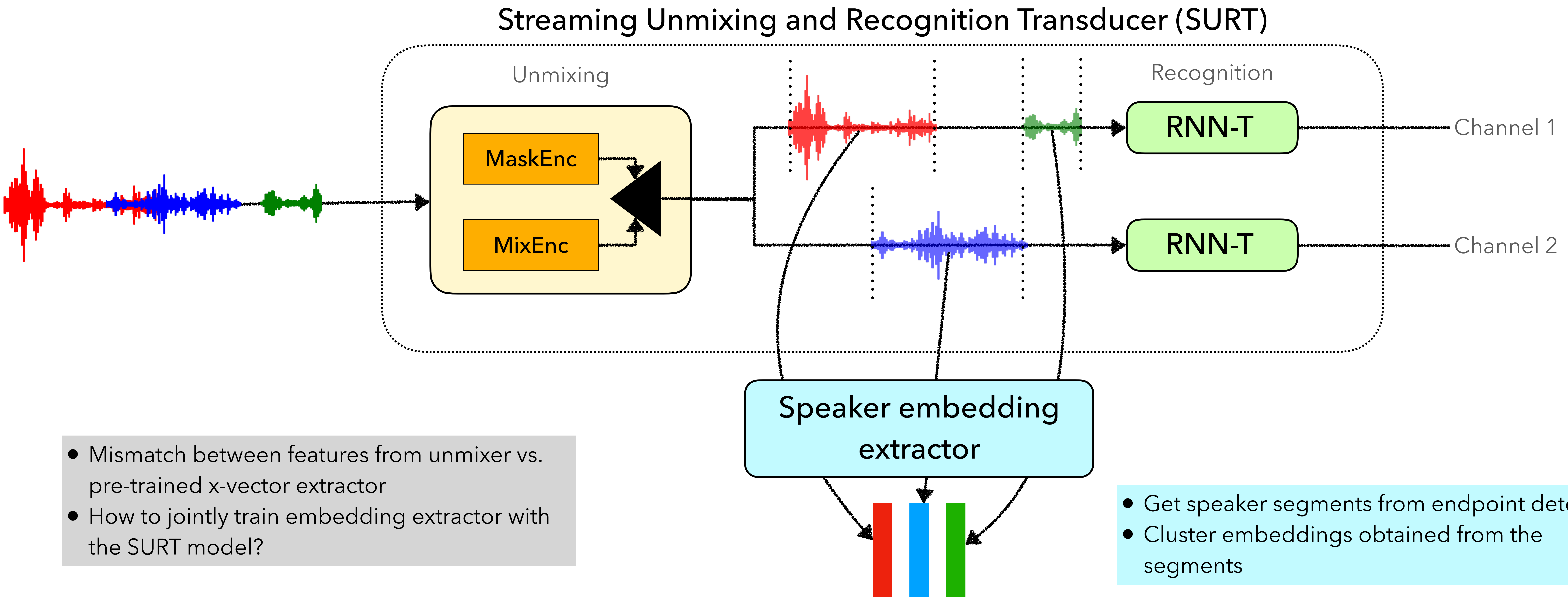
## Endpoint detection




- Predict <st> token to mark speaker turn changes
- Obtain timestamp of the <st> token from the lattice

# How to perform diarization with SURT?

## Speaker clustering




- Mismatch between features from unmixer vs. pre-trained x-vector extractor
- How to jointly train embedding extractor with the SURT model?

- Get speaker segments from endpoint detector
- Cluster embeddings obtained from the segments

# Neural MVDR beamforming

# Preliminary

## Mask-based MVDR beamforming

- Signal consists of a combination of target and distortion

$$\mathbf{Y}_{t,f} = \mathbf{d}_f \mathbf{S}_{t,f} + \mathbf{N}_{t,f}$$

- Here, $\mathbf{d}$ is called the steering vector

- A beamformer tries to **weight the sum of multi-channel signal** into enhanced signal

$$\hat{\mathbf{S}} = \mathbf{w}^H \mathbf{Y}, \quad \mathbf{w} \in \mathbb{C}^{D \times F}$$

- If weight of frequency bin is constant for all time steps, called time-invariant

# Preliminary

## Mask-based MVDR beamforming

- MVDR beamformer: minimum variance distortionless response
- Minimize the power of the interfering signal while preserving the distortionless source signal

$$
\begin{aligned}
\mathbf{w}_{\mathrm{MVDR}}(f) = \underset{\mathbf{w}}{\arg\min}\, & \mathbf{w}^{\mathrm{H}}(f)\Phi_{\mathbf{YY}}(f)\mathbf{w}(f) \\
\text{s.t.} \quad & \mathbf{w}(f)^{\mathrm{H}}\mathbf{d}(f) = 1
\end{aligned}
$$

- Here, $\Phi_{YY}(f)$ is the covariance of the noisy STFT at frequency $f$.

# Preliminary

## Mask-based MVDR beamforming

$$\gamma_{t,f,k}$$

Target mask

$$\boldsymbol{\Phi}_k(f) = \frac{1}{T}\sum_t \gamma_{t,f,k}\tilde{\mathbf{Y}}_{t,f}\tilde{\mathbf{Y}}_{t,f}^H$$

Spatial covariance matrix for target

$$\gamma_{t,f,n} = \sum_{k' \neq k} \gamma_{t,f,k'}$$

Distortion mask

$$\boldsymbol{\Phi}_n(f) = \frac{1}{T}\sum_t \gamma_{t,f,n}\tilde{\mathbf{Y}}_{t,f}\tilde{\mathbf{Y}}_{t,f}^H$$

Spatial covariance matrix for noise

**Beamforming vector**: gives the weight assigned to each channel for every frequency

$$\mathbf{w}_k(f) = \frac{\boldsymbol{\Phi}_n^{-1}(f)\boldsymbol{\Phi}_k(f)\mathbf{e}_{\mathrm{ref}}}{\mathrm{tr}\left(\boldsymbol{\Phi}_n^{-1}(f)\boldsymbol{\Phi}_k(f)\right)}$$

$$\hat{\mathbf{X}}_{t,f,k} = \tilde{\mathbf{Y}}_{t,f} \cdot \mathbf{w}_k(f)$$

Estimated signal for target $k$

# ADL-MVDR

## All deep learning MVDR

- Let us re-write the MVDR solution using the steering vector $\mathbf{d}_f$

$$\mathbf{w}_k(f) = \frac{\mathbf{\Phi}_n^{-1}(f)\mathbf{d}_f}{\mathbf{d}_f^H \mathbf{\Phi}_n^{-1}(f)\mathbf{d}_f}$$

- So we mainly need to estimate $\Phi_n^{-1}$ and $\mathbf{d}$ for each T-F bin. This can be done using neural networks (specifically, GRU-nets)

$$\mathbf{d}_{t,f} = \mathrm{GRUnet}(\Phi_k(t, f))$$

$$\Phi_k^{-1}(t, f) = \mathrm{GRUnet}(\Phi_k(t, f))$$

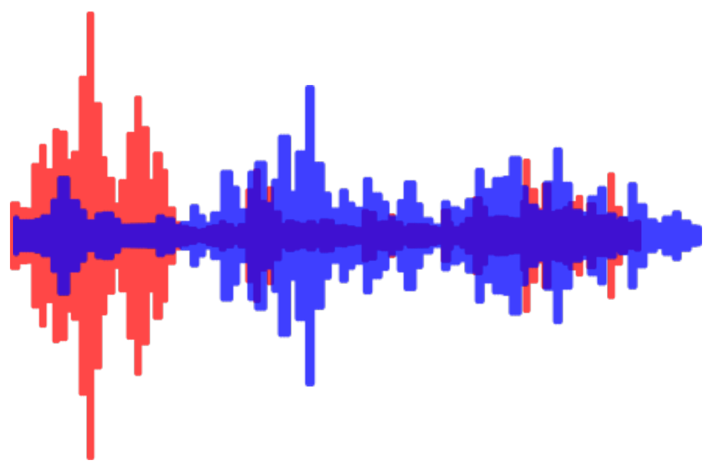
Zhang, Z., Yoshioka, T., Kanda, N., Chen, Z., Wang, X., Wang, D., & Eskimez, S.E. (2021). All-neural beamformer for continuous speech separation. *ArXiv, abs/2110.06428*.

# Graph-PIT for training SURT models

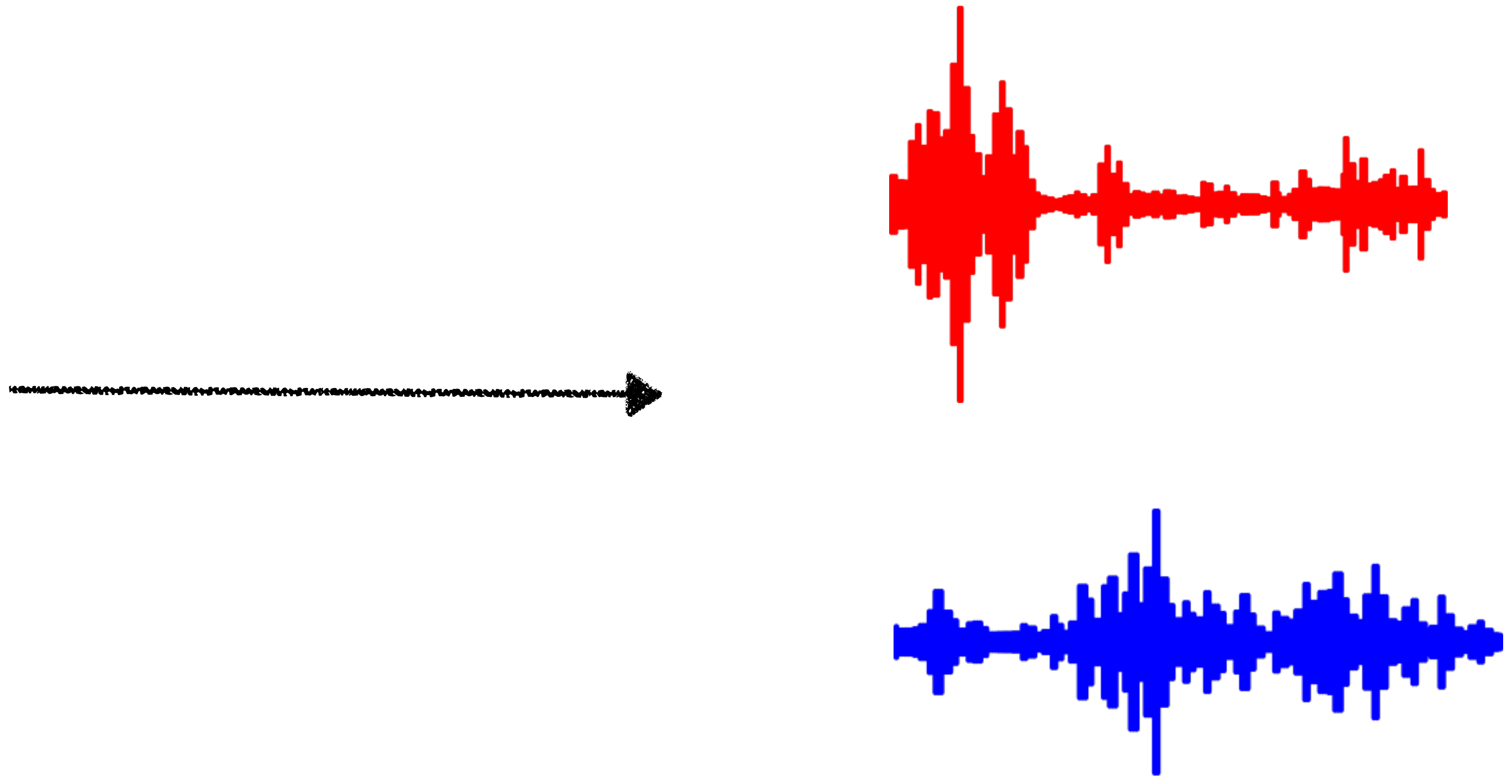
# Preliminary

## Continuous speech separation

- Speech separation using neural networks works well for fixed number of speakers, e.g., separating short 2-speaker mixtures



Input mixed speech



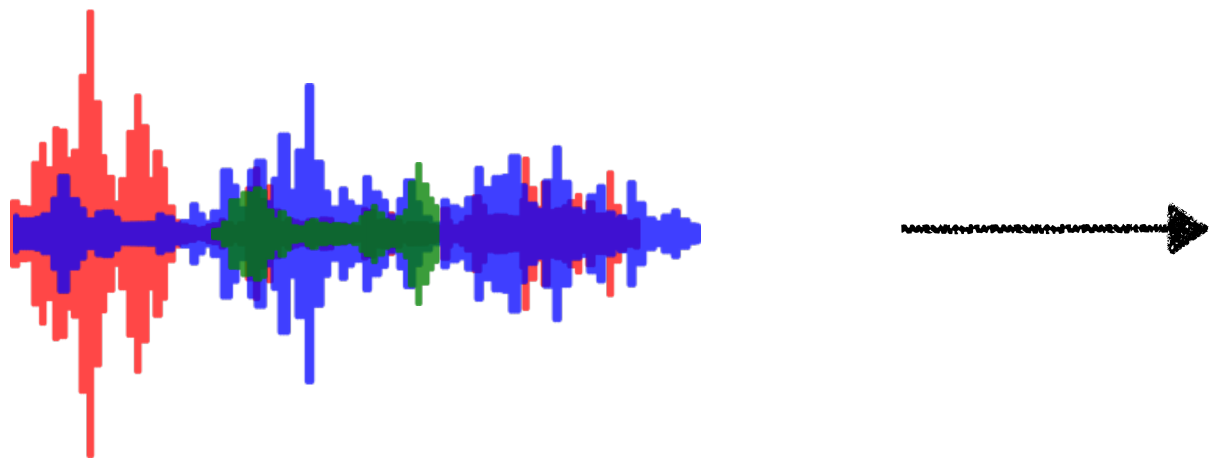
Separated speech

# Preliminary

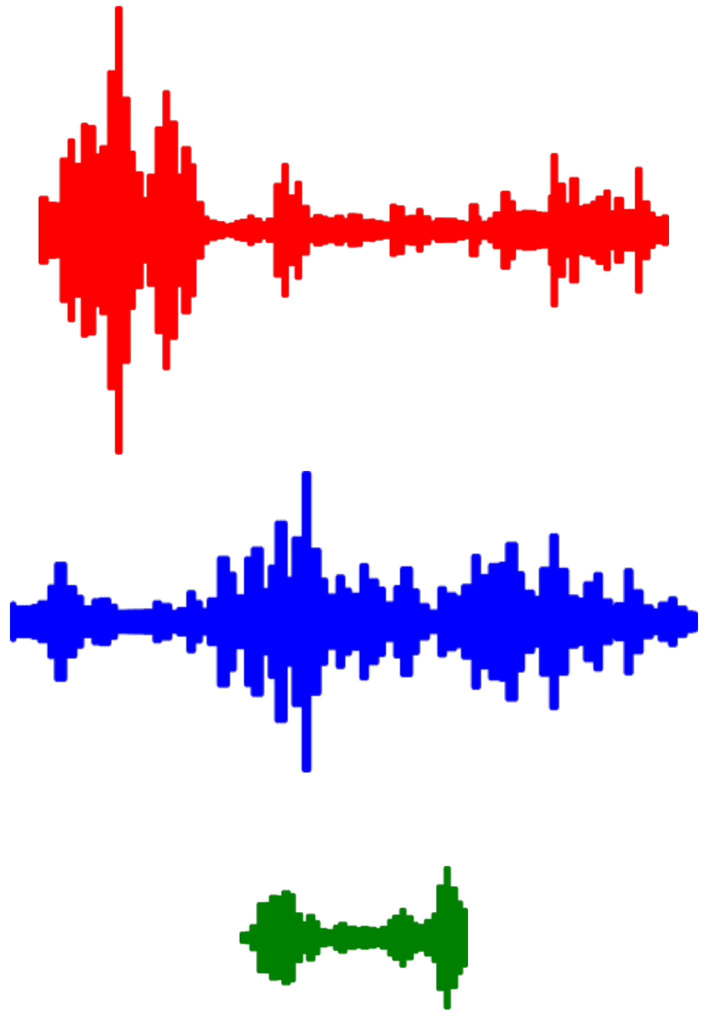
## Continuous speech separation

- But what about arbitrary number of speakers? **Problem:** neural networks are trained with fixed number of outputs

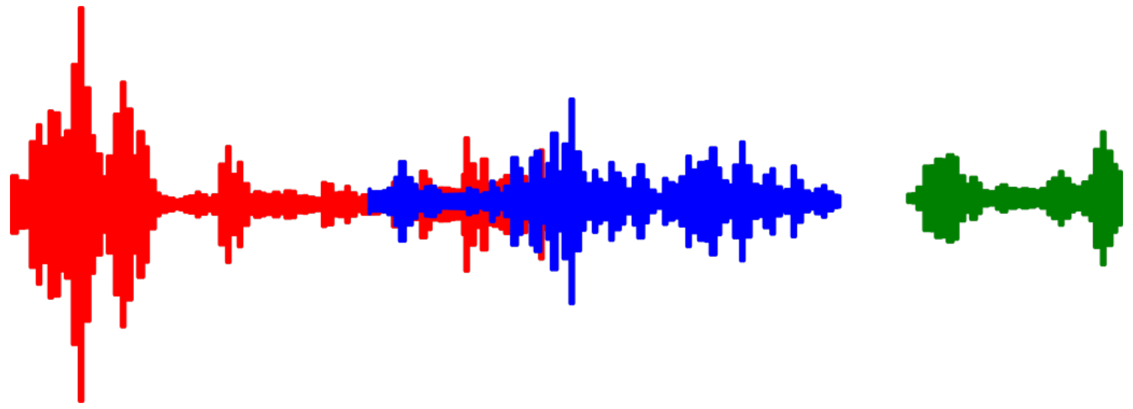
- Or long-form recordings? **Problem:** OOM



Mixed speech with 3 speakers



Separated speech

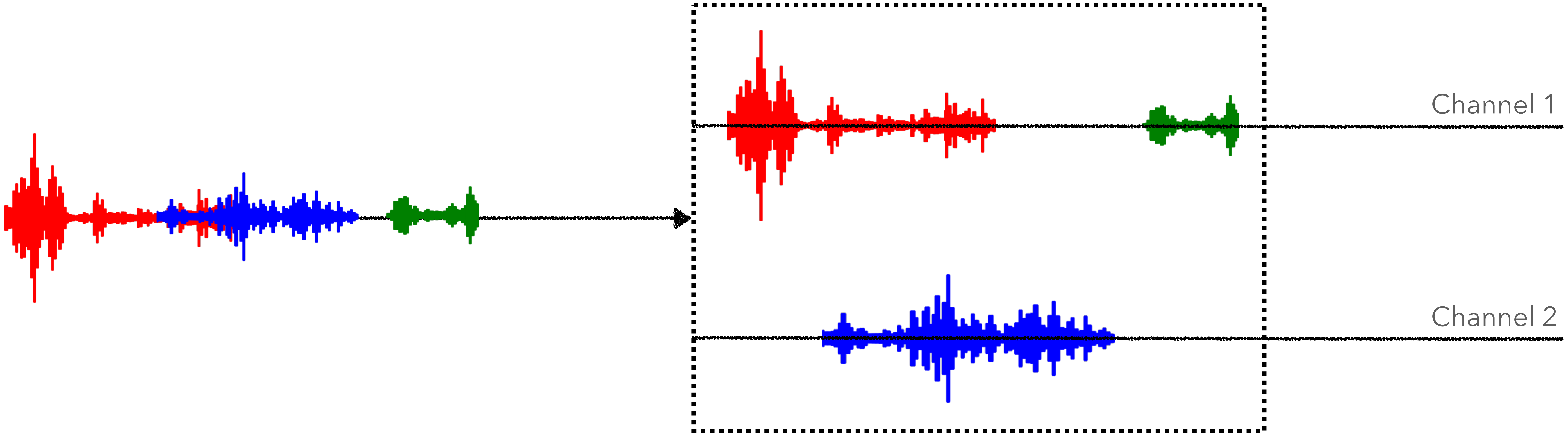


Long recording containing 3 speakers

# Preliminary

## Continuous speech separation

- Assumption: A small segment (say 2-3 seconds) will contain at most 2 speakers
- Separate small chunks into fixed number of outputs and **stitch**



# Preliminary

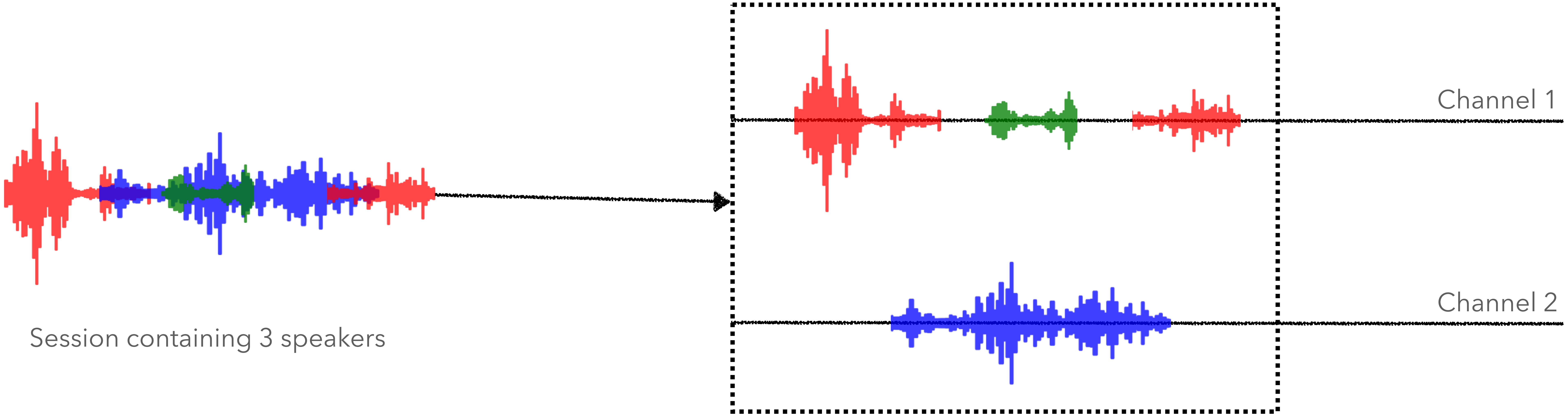
## Continuous speech separation

- Assumption: A small segment (say 2-3 seconds) will contain at most 2 speakers
- Trained with permutation invariant training (PIT) loss
- Assumption may not hold in practice!
- **Weaker assumption:** at most 2 speakers *at any instant of time*

# Graph-PIT

## Generalizing PIT for long recordings

- **Weaker assumption:** at most 2 speakers *at any instant of time*
- Allows to train on longer sessions with multiple speakers, as long as this assumption holds

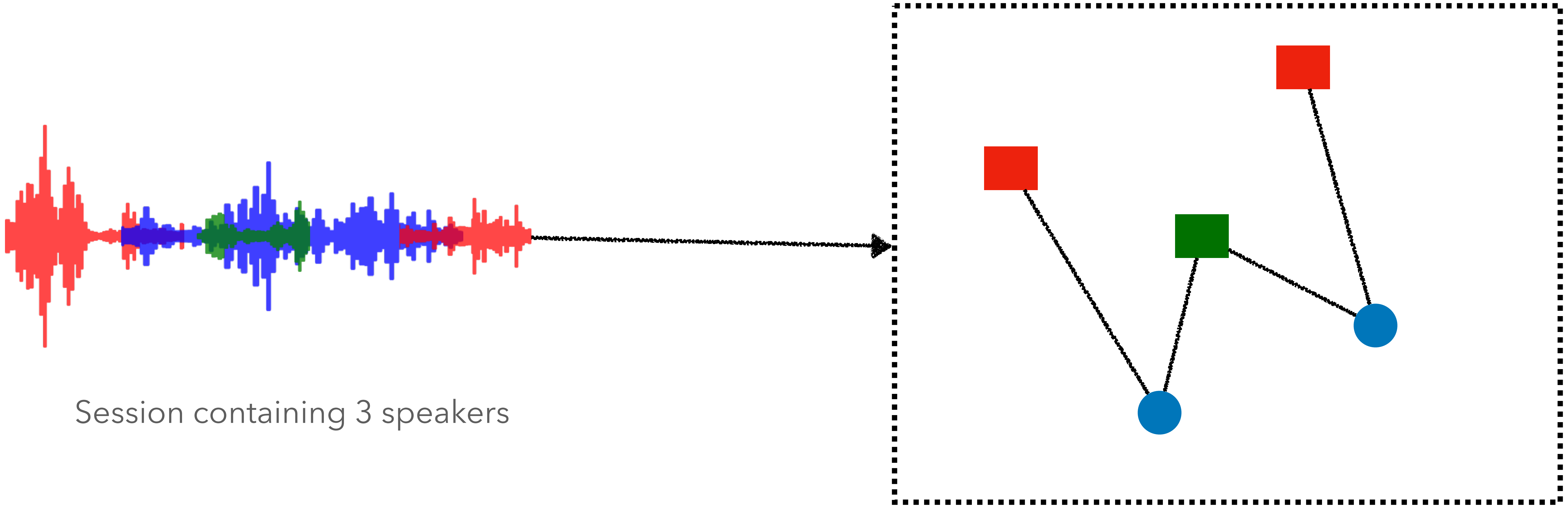


# Graph-PIT

## Generalizing PIT for long recordings

- Assign utterances to output channels such that overlapping utterances are on different channels

- Instance of **graph coloring problem**




- Each utterance is a node
- Overlapping utterances have an edge between them
- Color (here, shape) denotes assignment of utterance to channel

# Graph-PIT

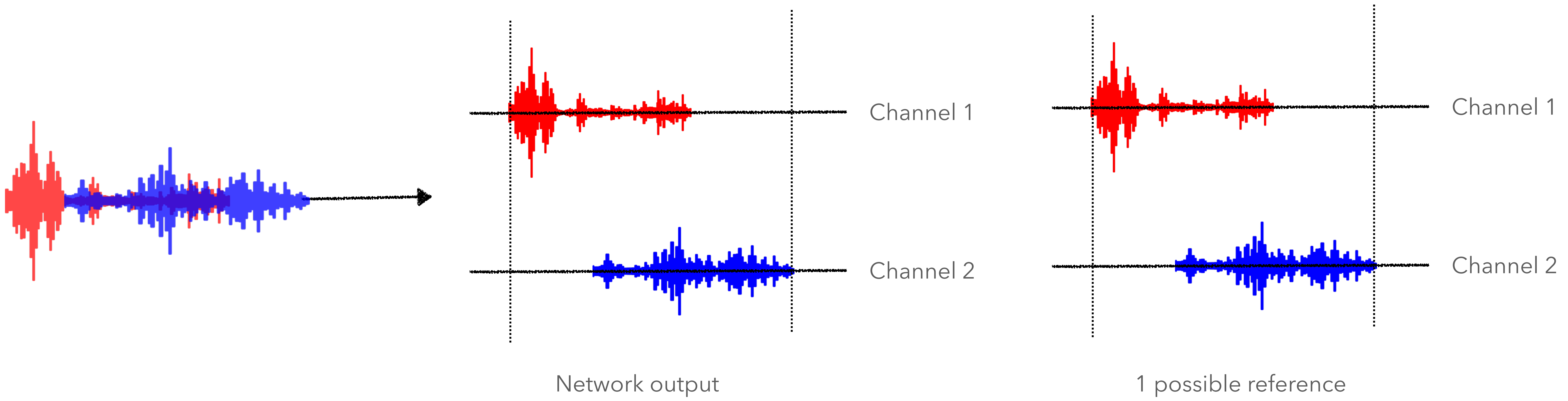
## Generalizing PIT for long recordings

- For training, minimize loss over all assignments
- Provides additional flexibility to the separation network, i.e., does not penalize network for correctly separating utterances
- **Problem:** Graph coloring is NP-hard!

# Graph-PIT

## Different types of losses

- “Aggregated” loss (e.g. a-SDR): aggregate over pairwise losses
- “Group” loss (e.g. sa-SDR): compute over the whole group



# Graph-PIT

## Different types of losses

- "Aggregated" loss (e.g. a-SDR): aggregate over pairwise losses
- "Group" loss (e.g. sa-SDR): compute over the whole group

$$\mathcal{L}^{\mathbf{a}-\mathrm{SDR}}(\mathbf{S}, \hat{\mathbf{S}}) = -\frac{1}{C}\sum_{c=1}^{C} 10\log_{10}\frac{\|\mathbf{s}_c\|^2}{\|\mathbf{s}_c - \hat{\mathbf{s}}_c\|^2}$$

$$= \frac{1}{C}\sum_{c=1}^{C}\left(-10\log_{10}\frac{\|\mathbf{s}_c\|^2}{\|\mathbf{s}_c - \hat{\mathbf{s}}_c\|^2}\right)$$

$$= \frac{1}{C}\sum_{c=1}^{C}\mathcal{L}^{\mathrm{SDR}}\left(\mathbf{s}_c, \hat{\mathbf{s}}_c\right)$$

$$\mathcal{L}^{\mathrm{sa-SDR}}(\mathbf{S}, \hat{\mathbf{S}}) = -10\log_{10}\frac{\sum_{c=1}^{C}\|\mathbf{s}_c\|^2}{\sum_{c=1}^{C}\|\mathbf{s}_c - \hat{\mathbf{s}}_c\|^2}$$

# Graph-PIT

## For the case of aggregated loss

- Compute matrix of pairwise losses $\mathbf{M}$
- Solve for best assignment using the Hungarian algorithm, $\mathcal{O}(C^3)$

$$
\begin{aligned}
\mathcal{L}^{\mathbf{a}-\mathrm{SDR}}(\mathbf{S}, \hat{\mathbf{S}}) &= -\frac{1}{C}\sum_{c=1}^{C} 10\log_{10}\frac{\|\mathbf{s}_c\|^2}{\|\mathbf{s}_c - \hat{\mathbf{s}}_c\|^2} \\
&= \frac{1}{C}\sum_{c=1}^{C}\left(-10\log_{10}\frac{\|\mathbf{s}_c\|^2}{\|\mathbf{s}_c - \hat{\mathbf{s}}_c\|^2}\right) \\
&= \frac{1}{C}\sum_{c=1}^{C}\boxed{\mathcal{L}^{\mathrm{SDR}}\left(\mathbf{s}_c, \hat{\mathbf{s}}_c\right)}
\end{aligned}
$$

Not defined when source is empty (often the case for CSS)

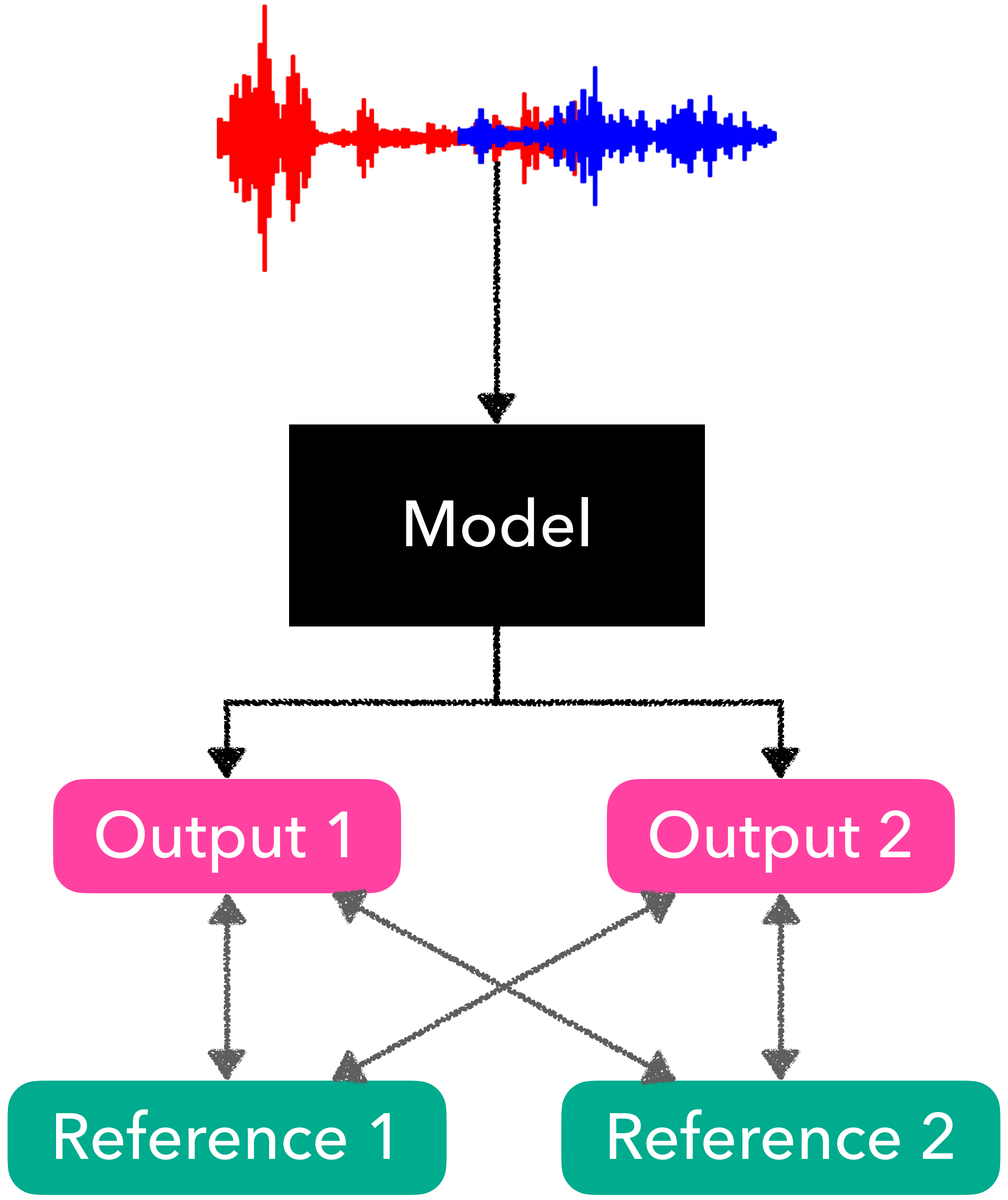
# Graph-PIT

## For the case of group loss

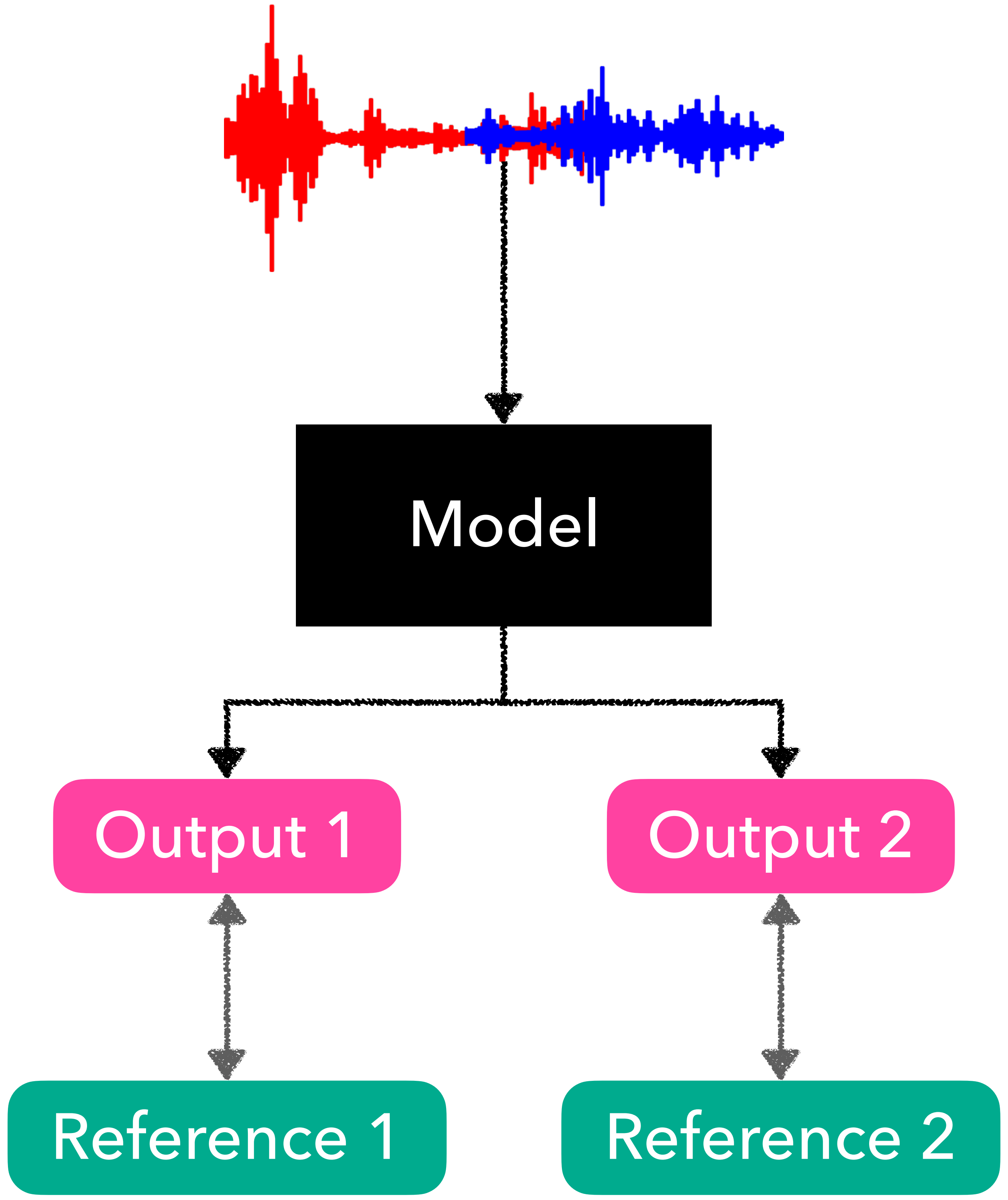
- Group loss (e.g., SA-SDR) is more suitable for training on long sessions, which can contain empty sources.
- We can still use Hungarian algorithm if we can decompose the loss into $\mathcal{J}^{\mathrm{uPIT}}(\hat{\mathbf{S}}, \mathbf{S}) = f(\min_{\mathbf{P} \in \mathcal{P}_C} \mathrm{Tr}(\mathbf{MP}, \hat{\mathbf{S}}, \mathbf{S}))$, where $f$ is a strictly monotonously increasing function.
- We can show that this is possible to do for SA-SDR loss, for example.

# Streaming Unmixing and Recognition Transducer

## PIT versus HEAT




Permutation invariant training (PIT)

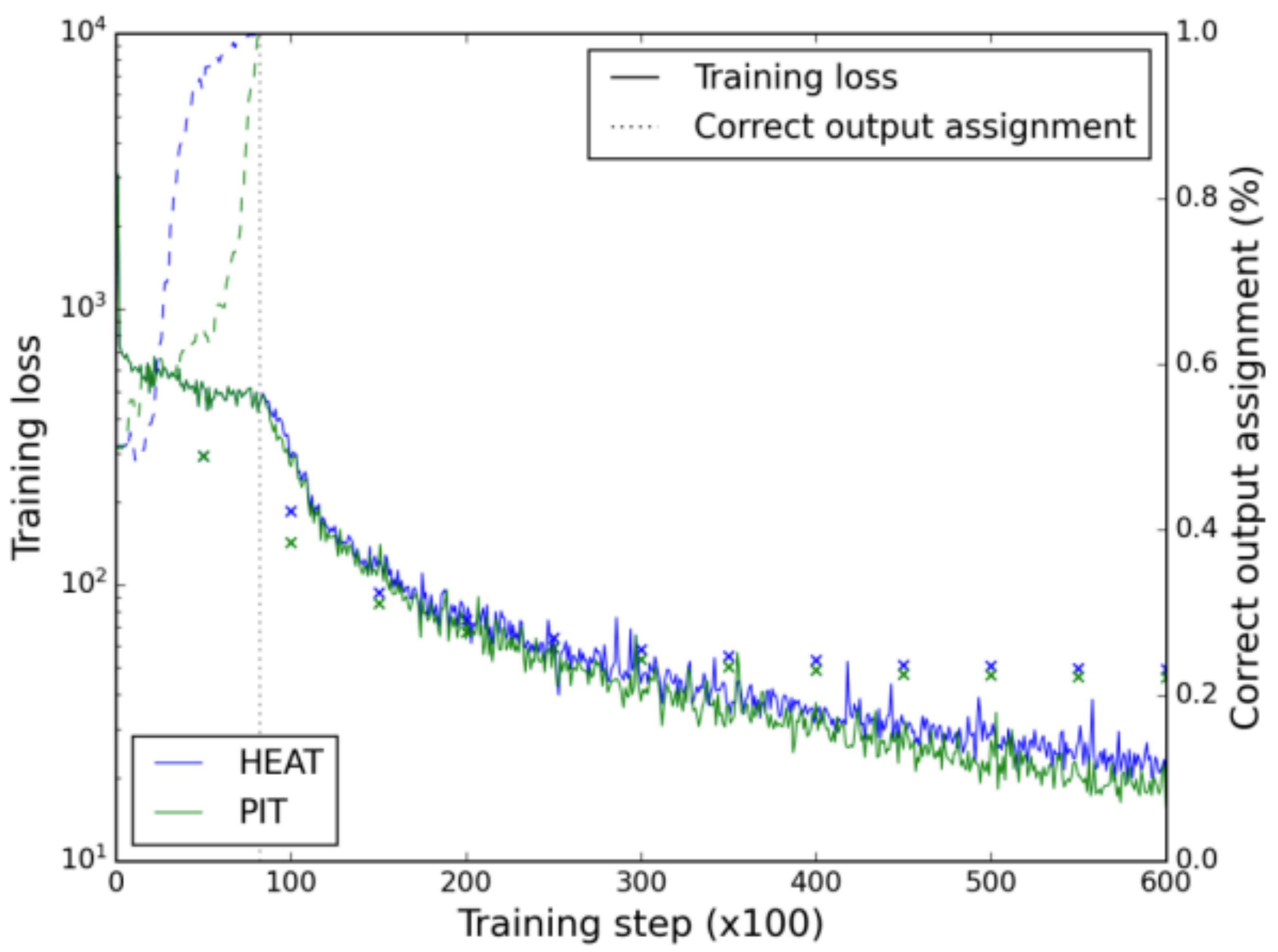



Heuristic error assignment training (HEAT)

# Streaming Unmixing and Recognition Transducer

## PIT versus HEAT

| Permutation invariant training (PIT) | Heuristic error assignment training (HEAT) |
|---|---|
| ✗ Requires computing all permutations of outputs and references | Requires computing only 1 permutation of output and reference ✓ |
| ✗ Can be prohibitively slow when N >> 2 (exponential in N) | Complexity increases linearly with N ✓ |




For utterances with non-zero delay, PIT learns the same heuristic as HEAT

# SURT objective

## Graph-PIT?

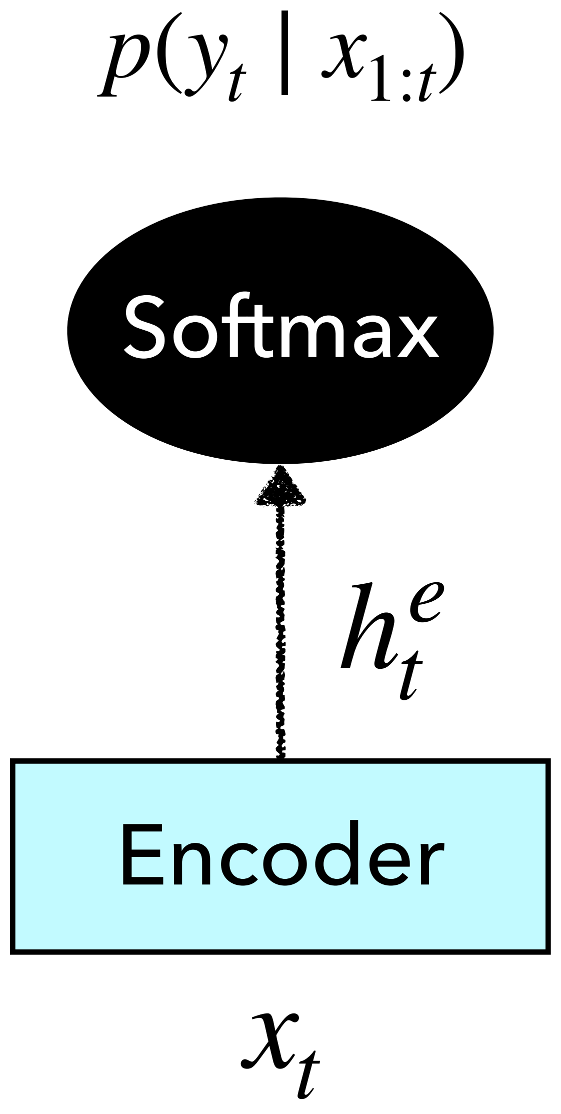
- HEAT is useful since it is feasible to train, rather than using PIT
- But it may be restraining, since we fix output assignment to channels
- Can we decompose underlying loss (RNN-T) such that we can use ideas from graph-PIT?

# RNN-Transducers

# Preliminary

## Connectionist Temporal Classification (CTC)

- Given input speech $\mathbf{X}$, find best word sequence $\mathbf{Y}$
- Need to compute $P(\mathbf{Y} \mid \mathbf{X})$
- For training, loss is $-\log P(\mathbf{Y} \mid \mathbf{X})$
- For inference, $\hat{\mathbf{Y}} = \mathrm{argmax}_{\mathbf{Y}} P(\mathbf{Y} \mid \mathbf{X})$




- **Problem:** Do not know alignment between $\mathbf{X}$ and $\mathbf{Y}$

# Preliminary

## Connectionist Temporal Classification (CTC)

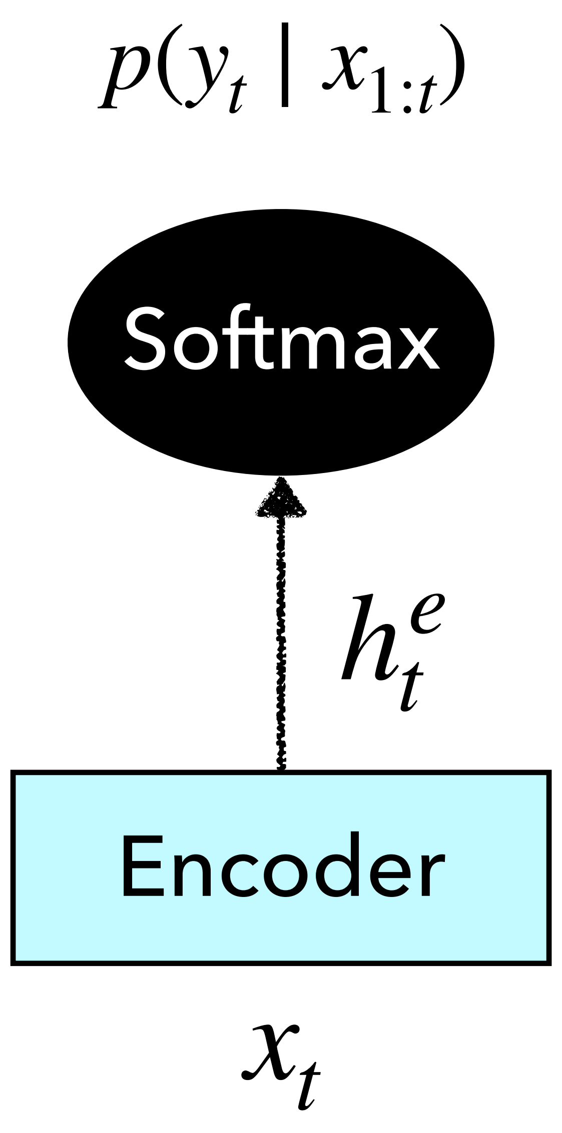
- **Problem:** Do not know alignment between $\mathbf{X}$ and $\mathbf{Y}$
- **Solution:** sum over all possible **alignments**

$$P(\mathbf{Y} \mid \mathbf{X}) = \sum_{A \in \mathcal{A}_{\mathbf{Y}}^{T}} P(A, \mathbf{Y} \mid \mathbf{X}) = \sum_{A \in \mathcal{A}_{\mathbf{Y}}^{T}} P(\mathbf{Y} \mid A, \mathbf{X}) P(A \mid \mathbf{X})$$

$$= \sum_{A \in \mathcal{A}_{\mathbf{Y}}^{T}} P(\mathbf{Y} \mid A) P(A \mid \mathbf{X}) = \sum_{A \in \mathcal{A}_{\mathbf{Y}}^{T}} P(A \mid \mathbf{X})$$

$$= \sum_{A \in \mathcal{A}_{\mathbf{Y}}^{T}} \boxed{\prod_{t=1}^{T} P(a_t \mid \mathbf{X})}$$

Conditional independence of outputs



# Preliminary

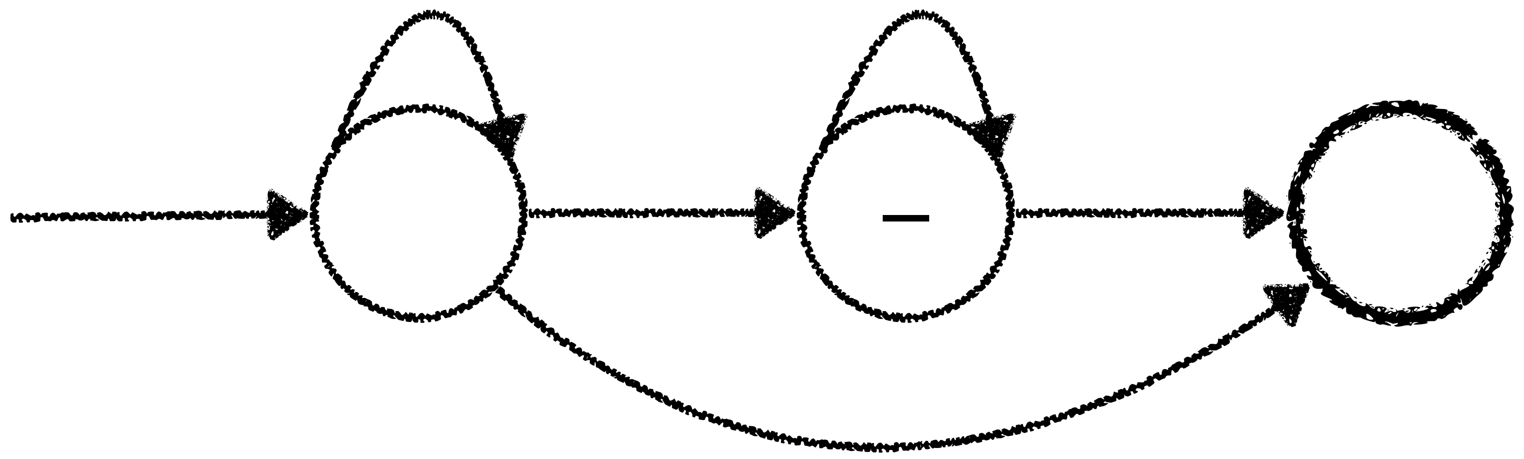
## Connectionist Temporal Classification (CTC)

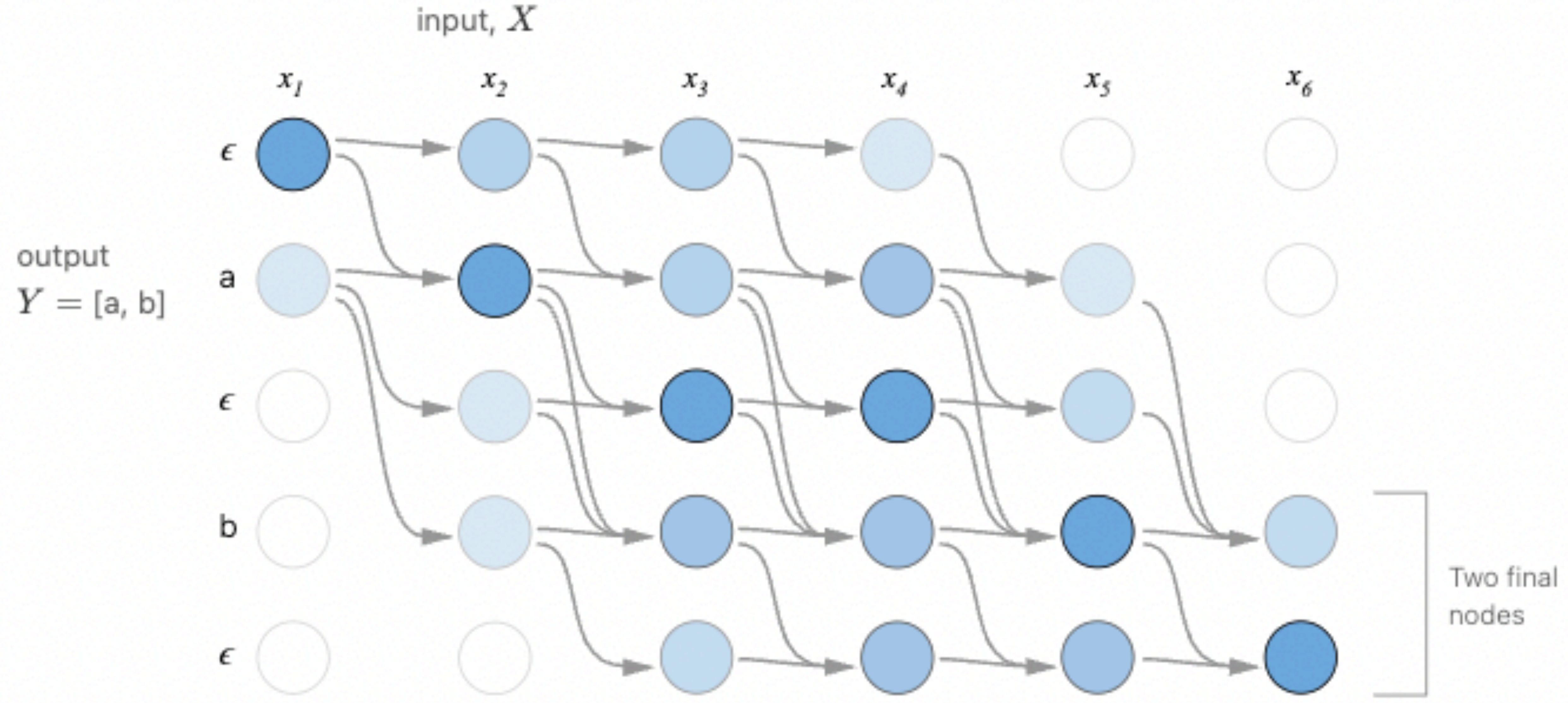
- What is an **alignment**?
- Example: $\mathbf{X}$ is of length 5, $\mathbf{Y}$ is CAT
- Alignments: CCAAT, $\epsilon$CATT, CAA$\epsilon$T, etc.
- To get word from alignment, first collapse repetitions, then remove $\epsilon$
- Now we only need a way to sum over all such alignments
- **Problem:** Exponentially many alignments

# Preliminary

## Connectionist Temporal Classification (CTC)

- **Problem:** Exponentially many alignments
- **Solution:** dynamic programming
- Similar to HMM forward algorithm




Node $(s, t)$ in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after $t$ input steps.

https://distill.pub/2017/ctc/

# Preliminary

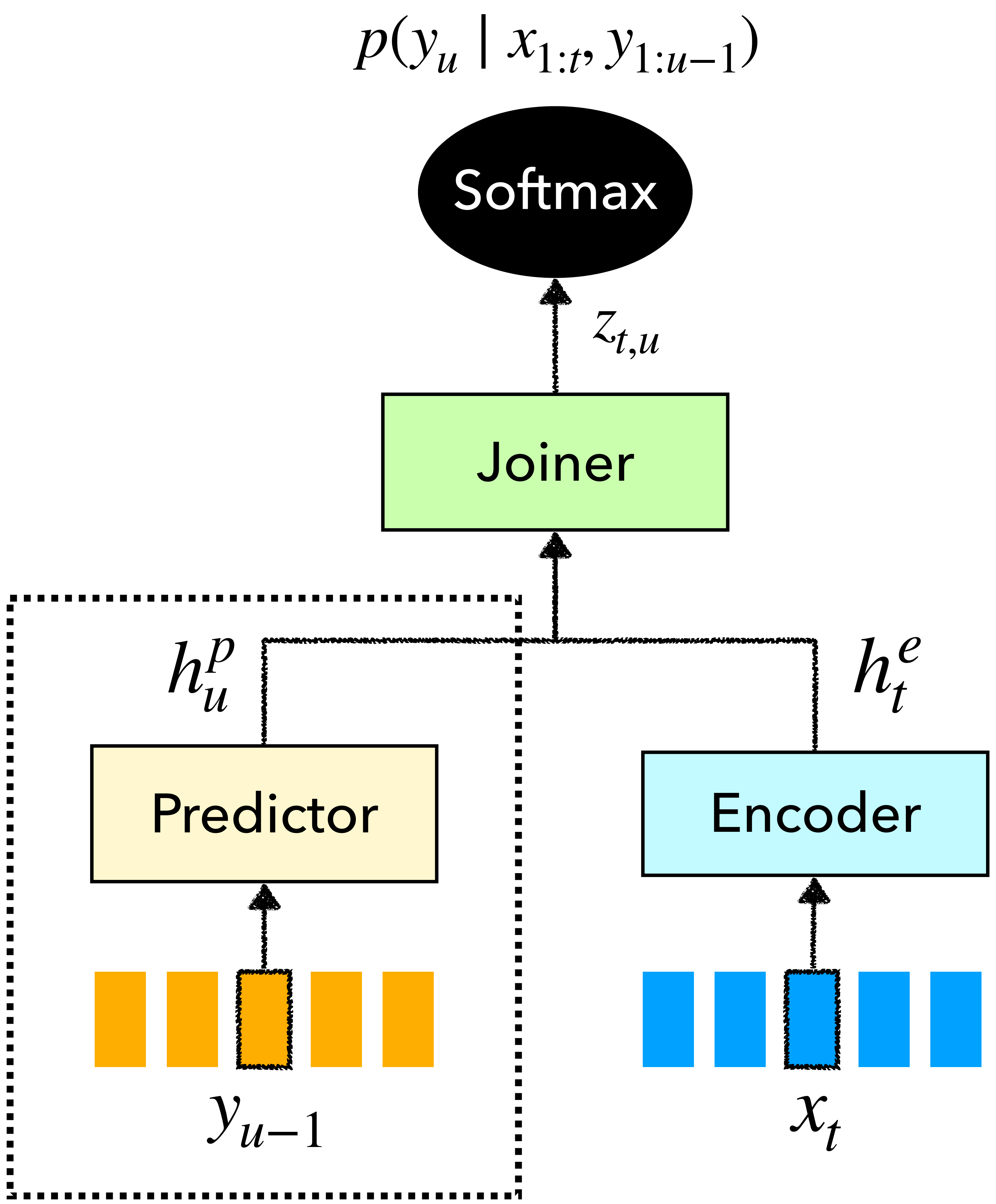
## Problems with CTC

1. Conditional independence of outputs
2. Output sequence must be shorter than input sequence

# RNN-Transducer

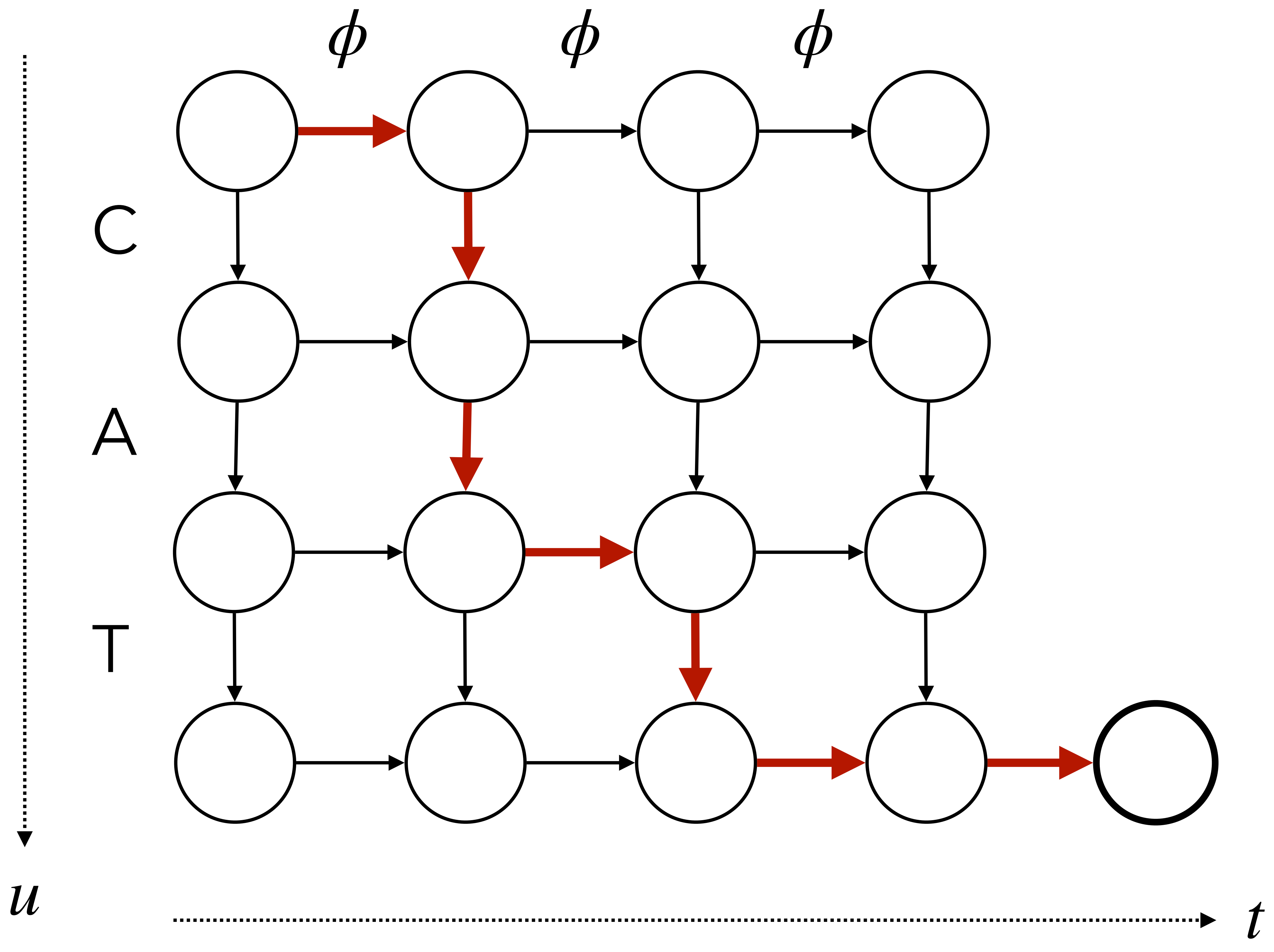
## Solves both of the problems with CTC

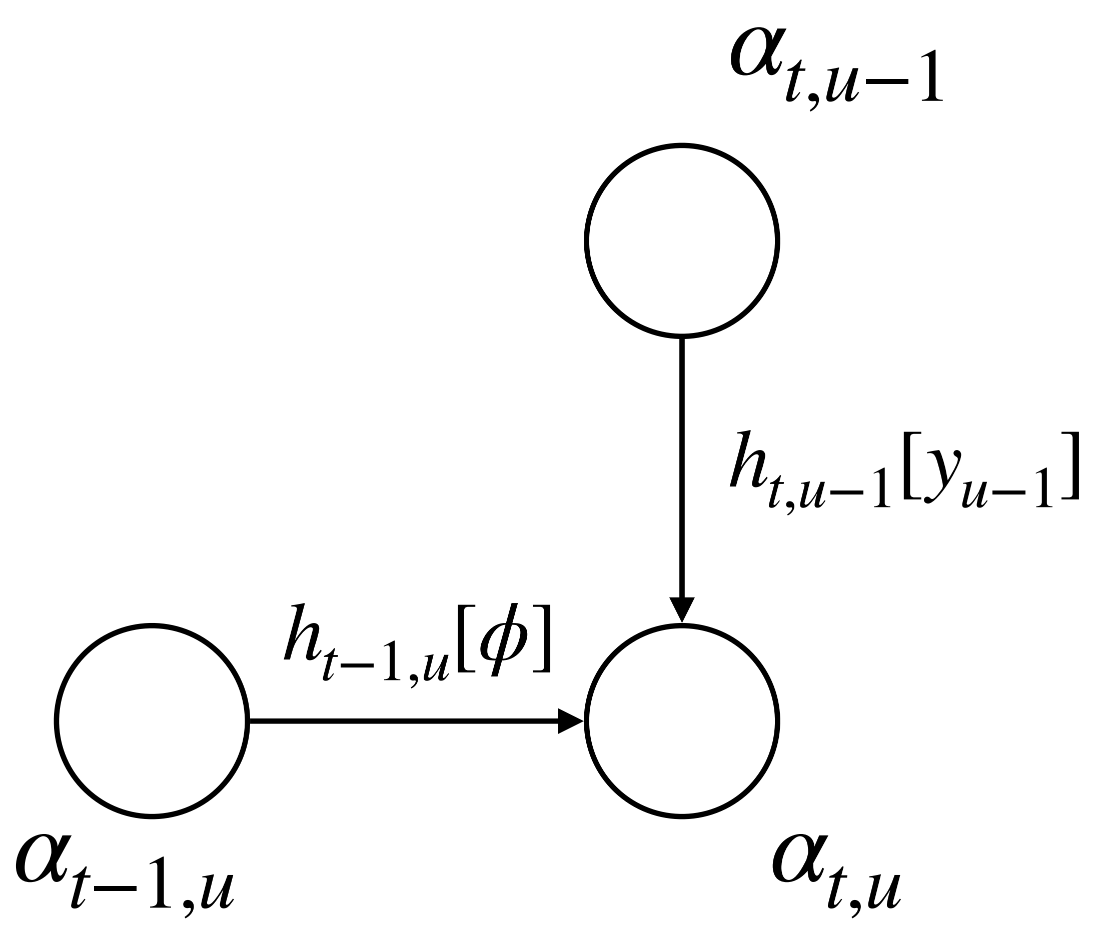
1. Conditional independence of outputs
    - *Use a predictor network (autoregressive model on previous outputs)*
2. Output sequence must be shorter than input sequence
    - *Allow multiple outputs at each time step*



# RNN-Transducer

## Monotonic alignments




**Forward algorithm**

# RNN-Transducer

## Inference: greedy decoding

- Always pick the top output at each time step
- If $\phi$ is generated, move to the next time step
- Otherwise, stay in the same time step and generate next label

- **Problem**: we do not want to stay in time frame $t$ forever

# RNN-Transducer

## Inference: beam search decoding

- **Problem**: we do not want to stay in time frame $t$ forever
- Keep track of 2 sets of hypotheses: A and B
  - A: set of hypotheses starting with non-black symbol, i.e., at time $t + 1$
  - B: set of hypotheses starting with blank, i.e., at time $t$
- Exit from $t$ if B has W (beam size) hypotheses more probable than best hypothesis in A

Expand current time step until we have W hypotheses

- Move to $t + 1$; empty A and move all B into A

# RNN-Transducer

## Inference: WFST decoding with k2

- Constrain number of outputs to at most $S$ symbol per frame; after this, force transition to next time step
- For WFST decoding, $S = 1$, similar to hybrid or CTC decoding
- Also use Conv1D instead of LSTM in prediction network
- This allows us to use WFSTs for decoding since the number of decoder states is now finite.
- The FSA beam search algorithm generates a lattice, after which the highest probability label sequence can be searched in the lattice.

# RNN-Transducer

## Other optimizations

- RNN-T is a memory hungry model; loss computation needs to store BxTxUxV logits (~4-5 G)
- Most log-prob mass is concentrated at only few tokens (say 5 tokens)
- Estimate which are these tokens using easy-to-compute method (simple addition of encoder and predictor representations)
- Then compute actual loss only for these 5 tokens: BxTxUx5

# Self-supervised learning in speech

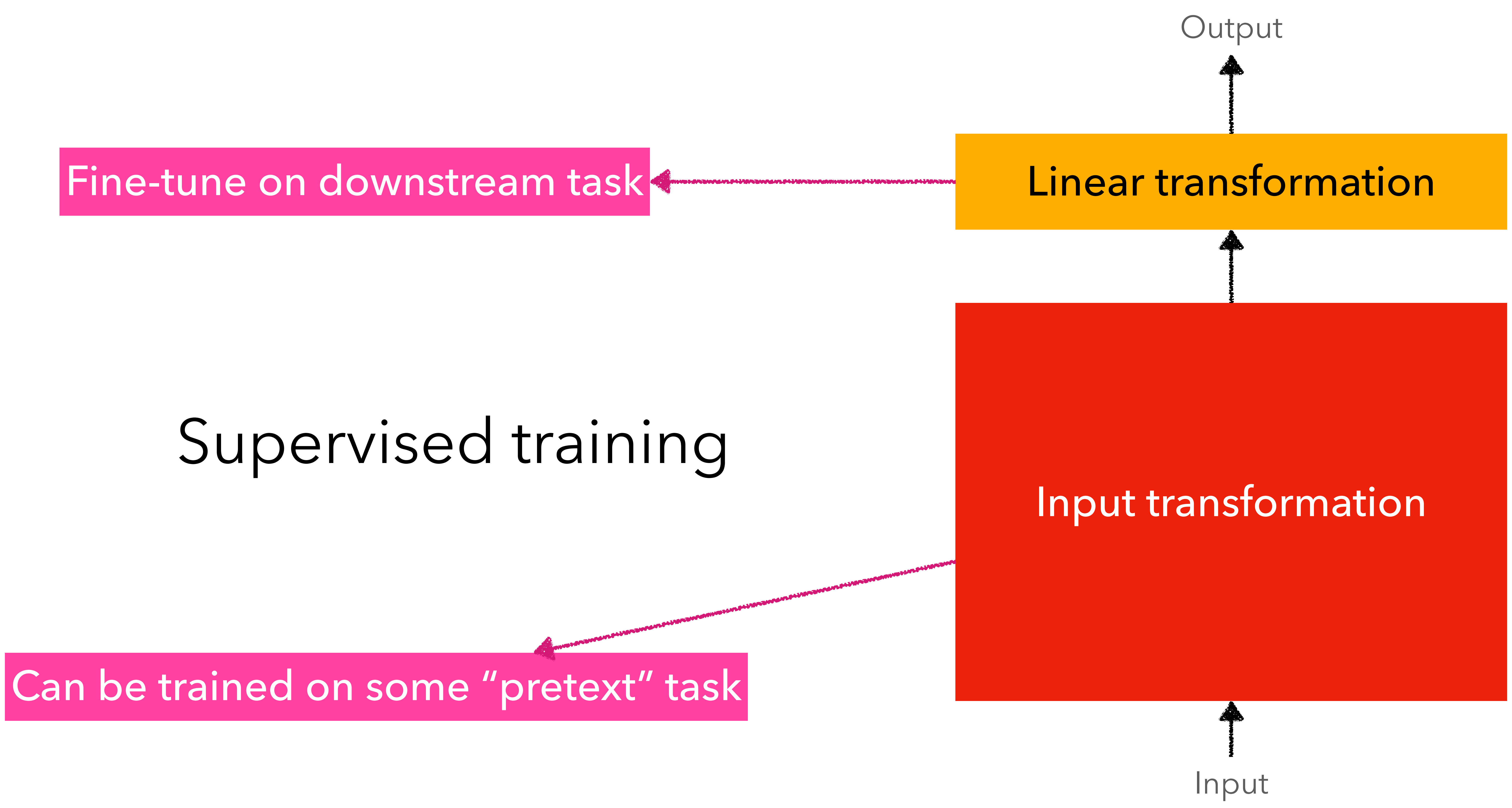
# Motivation

## From supervised to self-supervised

- Deep neural networks are good at learning from labeled data
- But not enough labeled data available (e.g. expensive to transcribe speech)
- Idea: pretext task and downstream task

# Motivation

## From supervised to self-supervised




Convolutional

Recurrent

Self-attention

# Pretext tasks

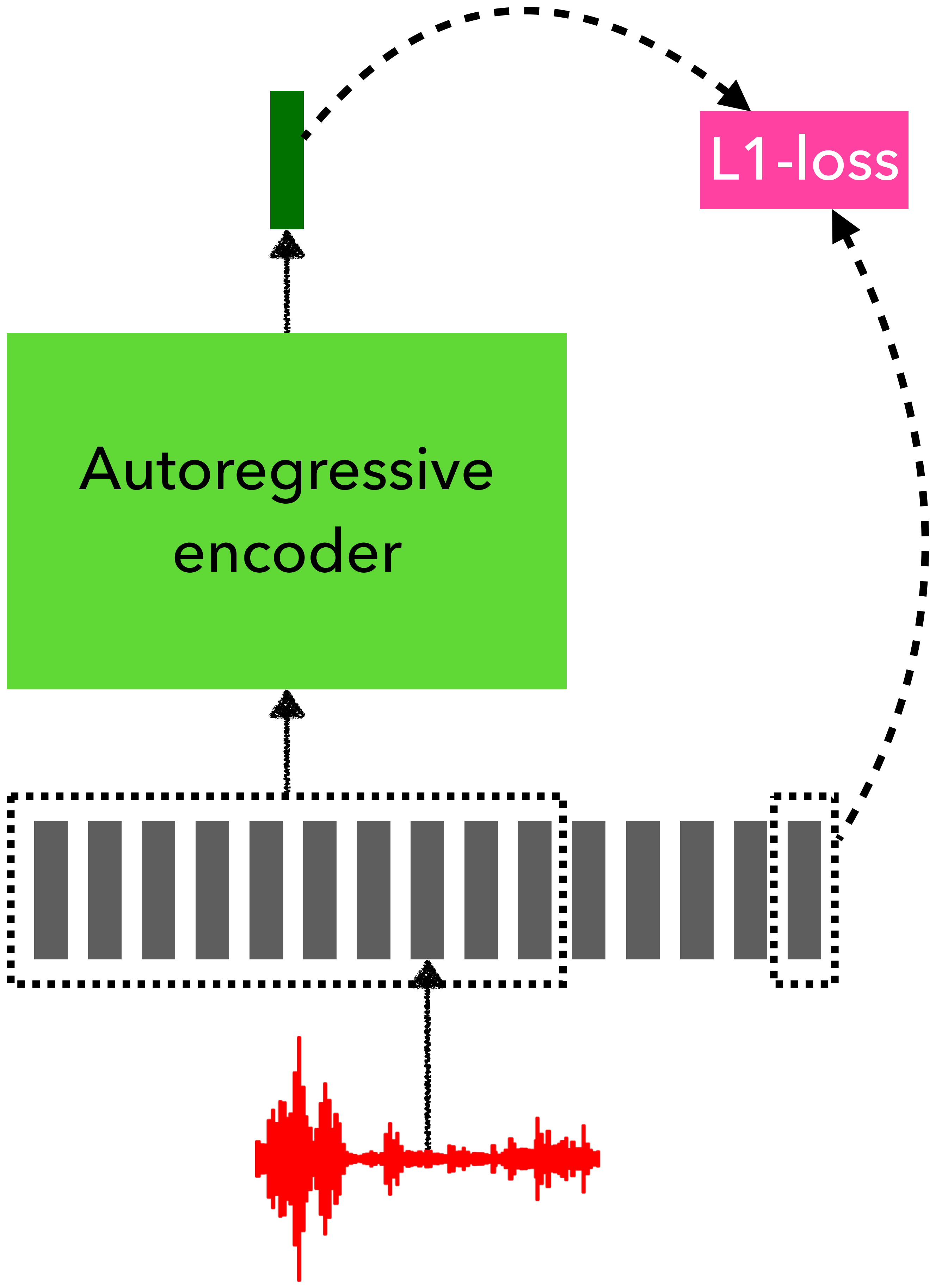
## Prediction-based and reconstruction-based

- **Prediction-based**
  - Predict future or masked tokens based on other tokens
  - E.g.: language modeling
- **Reconstruction-based**
  - Reconstruct clean input from noisy input
  - E.g.: denoising autoencoders

# Prediction-based training

## Autoregressive Predictive Coding (APC)

Chung, Yu-An and James R. Glass. “Generative Pre-Training for Speech with Autoregressive Predictive Coding.” *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020): 3497-3501.




Log-mel filterbanks

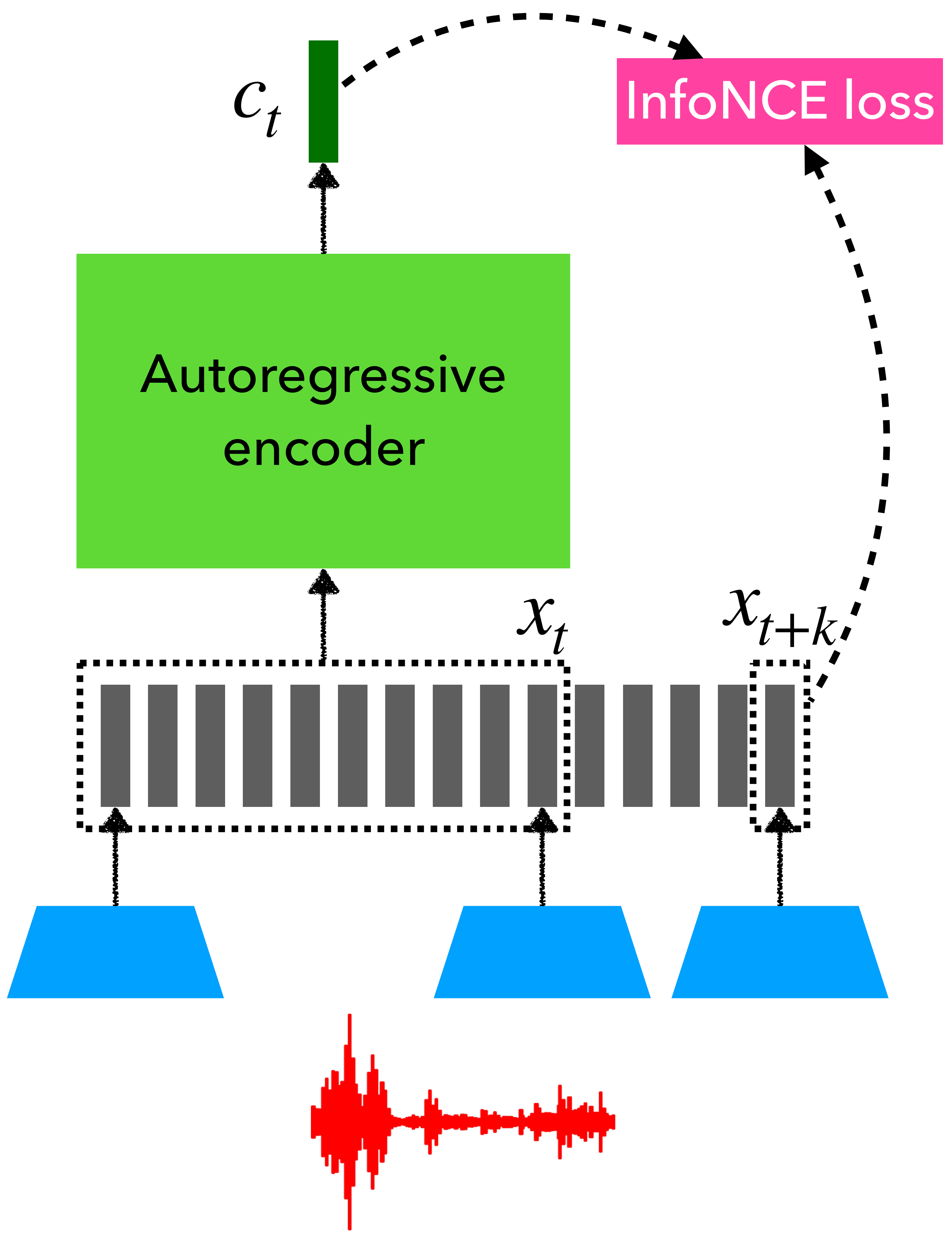
**Objective:** Predict the feature vector k steps into the future using an autoregressive model

**Loss function:** L1-loss

# Prediction-based training

## Contrastive Predictive Coding (CPC)

Oord, Aäron van den et al. “Representation Learning with Contrastive Predictive Coding.” *ArXiv* abs/1807.03748 (2018)




Latent-space features

Feature encoder (convolutional)

**Objective:** Predict future in *latent space* for next k steps using an autoregressive model

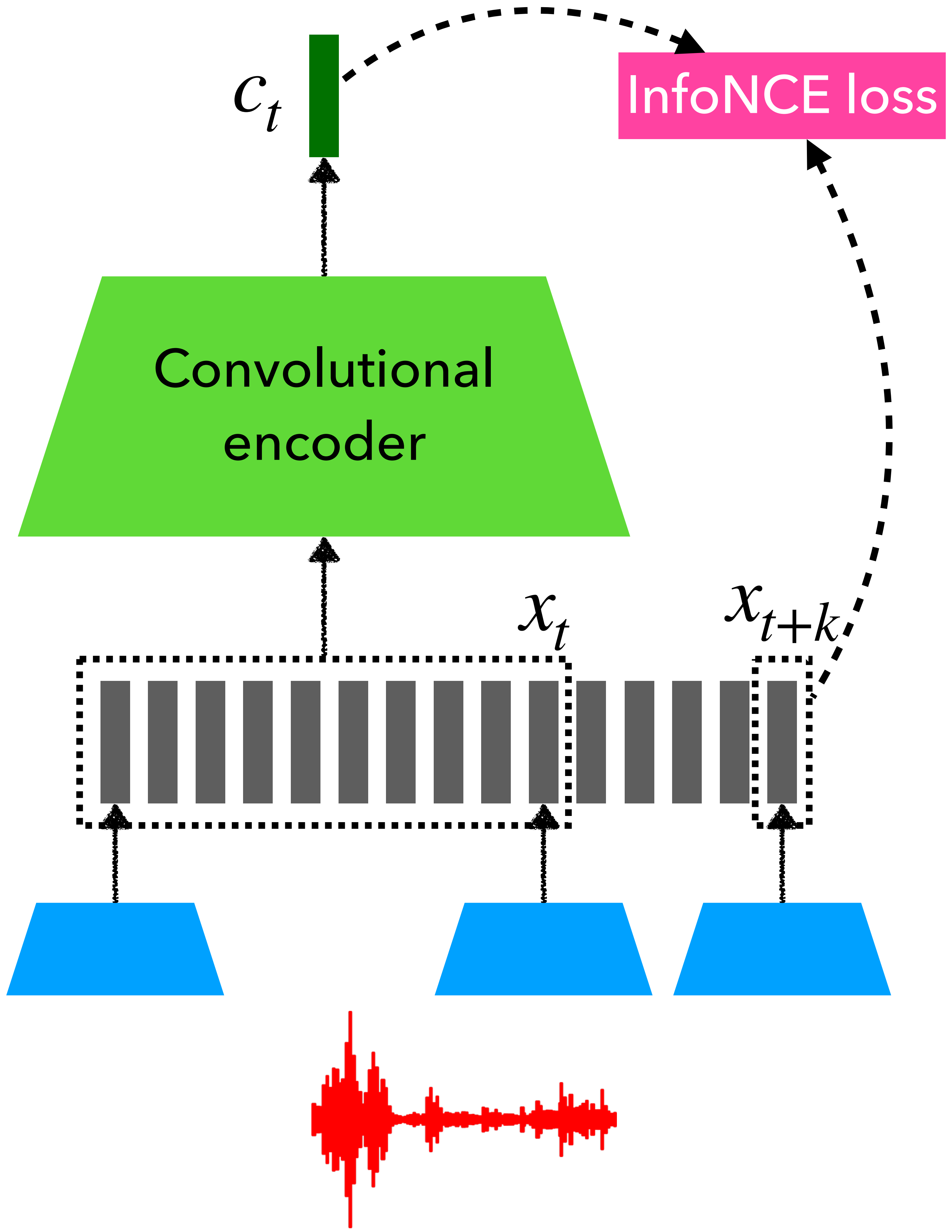
**Loss function:** InfoNCE loss

$$\mathcal{L}_N = -\mathbb{E}_X \left[ \log \frac{f_k\left(x_{t+k}, c_t\right)}{\sum_{x_j \in X} f_k\left(x_j, c_t\right)} \right]$$

# Prediction-based training

## Wav2Vec

Schneider, Steffen et al. "wav2vec: Unsupervised Pre-training for Speech Recognition." *INTERSPEECH* (2019).




**Objective:** Predict future in *latent space* for next k steps using an autoregressive model

**Loss function:** InfoNCE loss

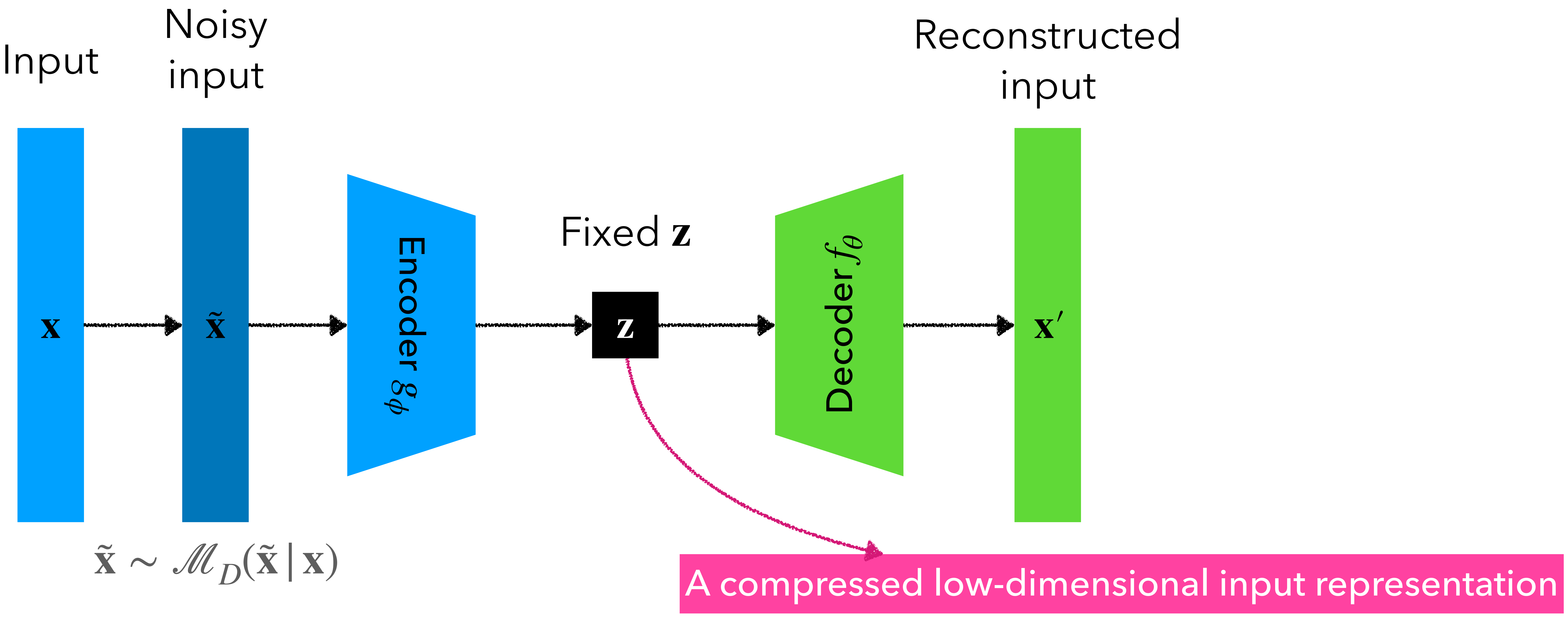
$$\mathcal{L}_N = -\mathbb{E}_X\left[\log \frac{f_k\left(x_{t+k}, c_t\right)}{\sum_{x_j \in X} f_k\left(x_j, c_t\right)}\right]$$

Latent-space features

Feature encoder (convolutional)

# Reconstruction-based training

## Denoising Autoencoders (DAE)

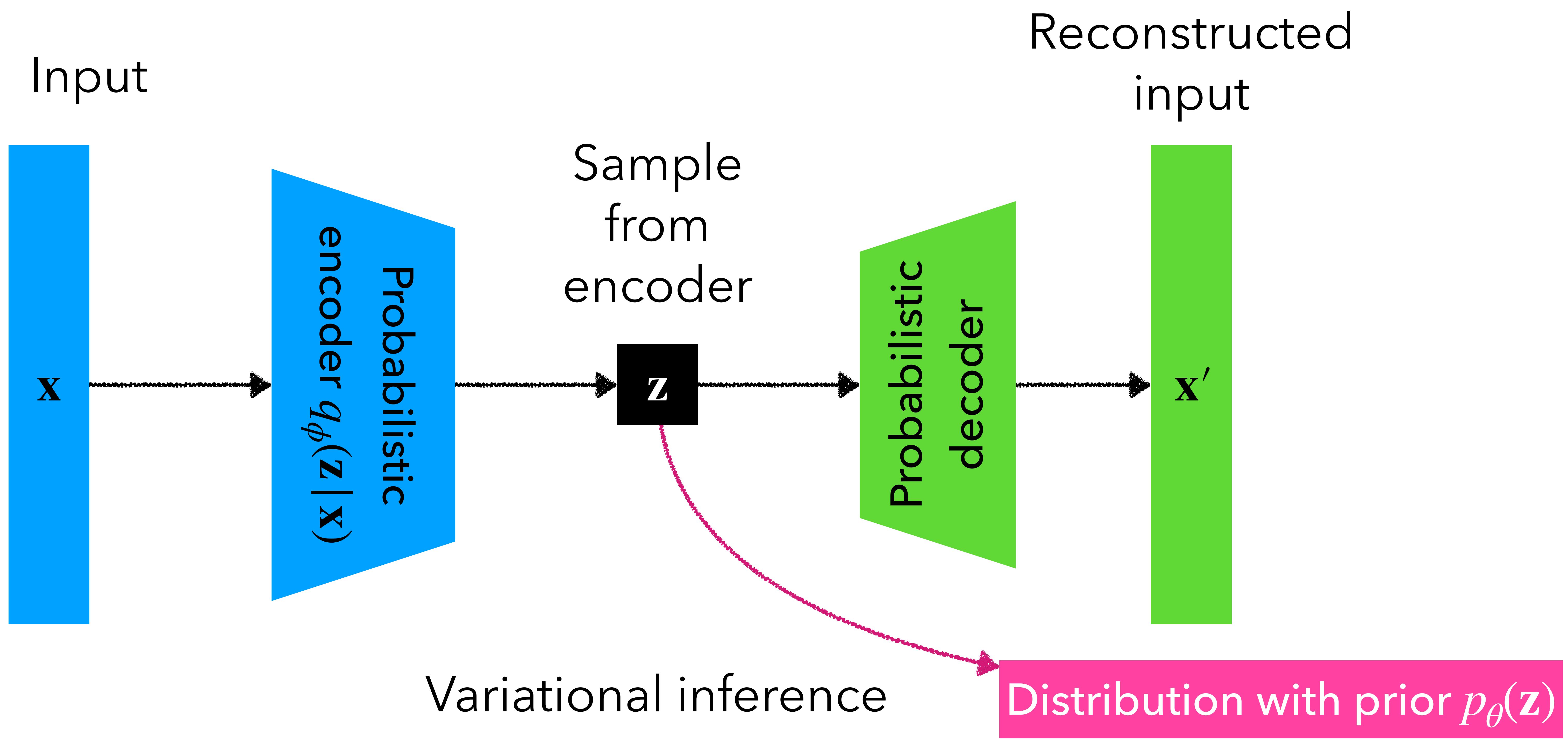



$$L_{\text{DAE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} \left( \mathbf{x}^{(i)} - f_\theta \left( g_\phi \left( \tilde{\mathbf{x}}^{(i)} \right) \right) \right)^2$$

# Reconstruction-based training

## Variational Autoencoders (VAE)

Kingma, Diederik P. and Max Welling. “Auto-Encoding Variational Bayes.” *CoRR* abs/1312.6114 (2014): n. pag.




Leads to posterior collapse in practice

# Reconstruction-based training

## Vector quantized Variational Autoencoders (VQ-VAE)

Oord, Aäron van den et al. “Neural Discrete Representation Learning.” *NIPS* (2017).

**Idea:** Learn a latent distribution on quantized input

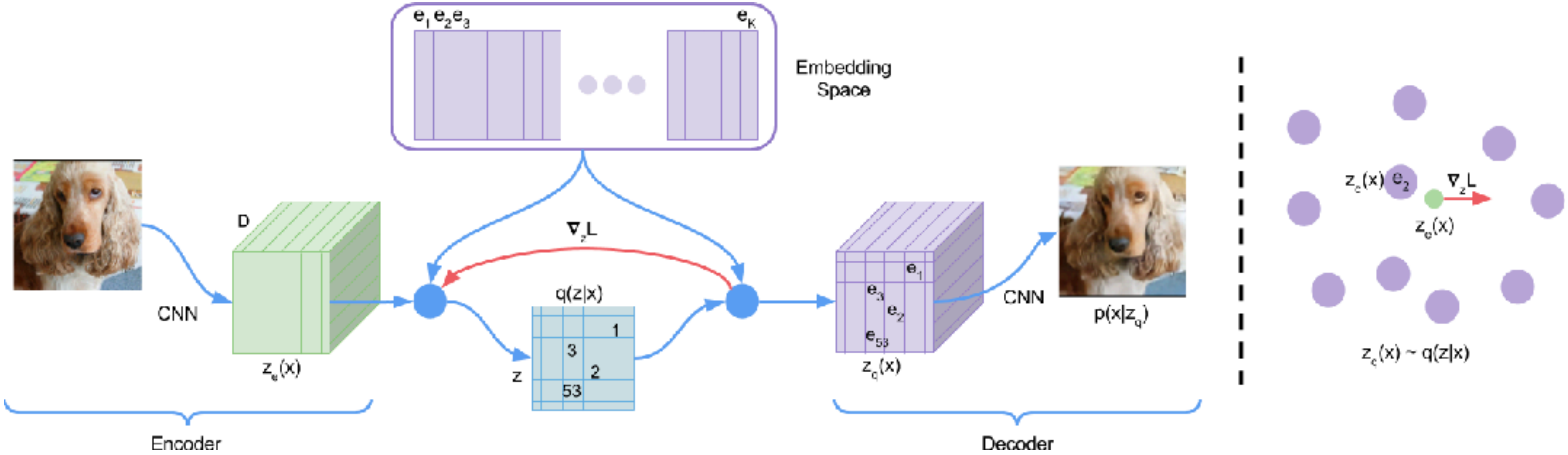
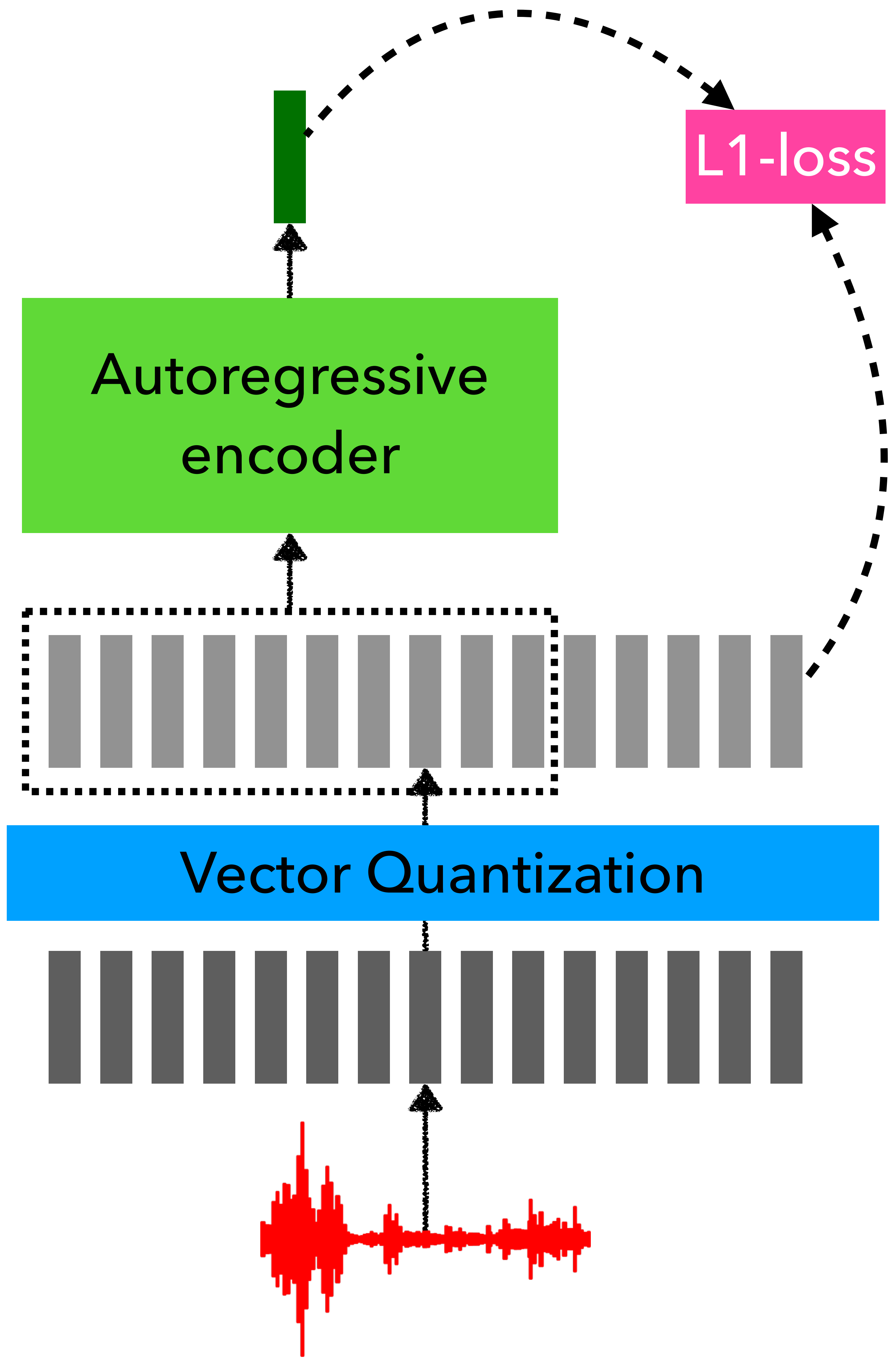



**Figure source:** Original paper from van den Oord et al. (2017)

# Prediction-based training (revisited)

## VQ-APC

Chung, Yu-An et al. “Vector-Quantized Autoregressive Predictive Coding.” *INTERSPEECH* (2020).




Log-mel filterbanks

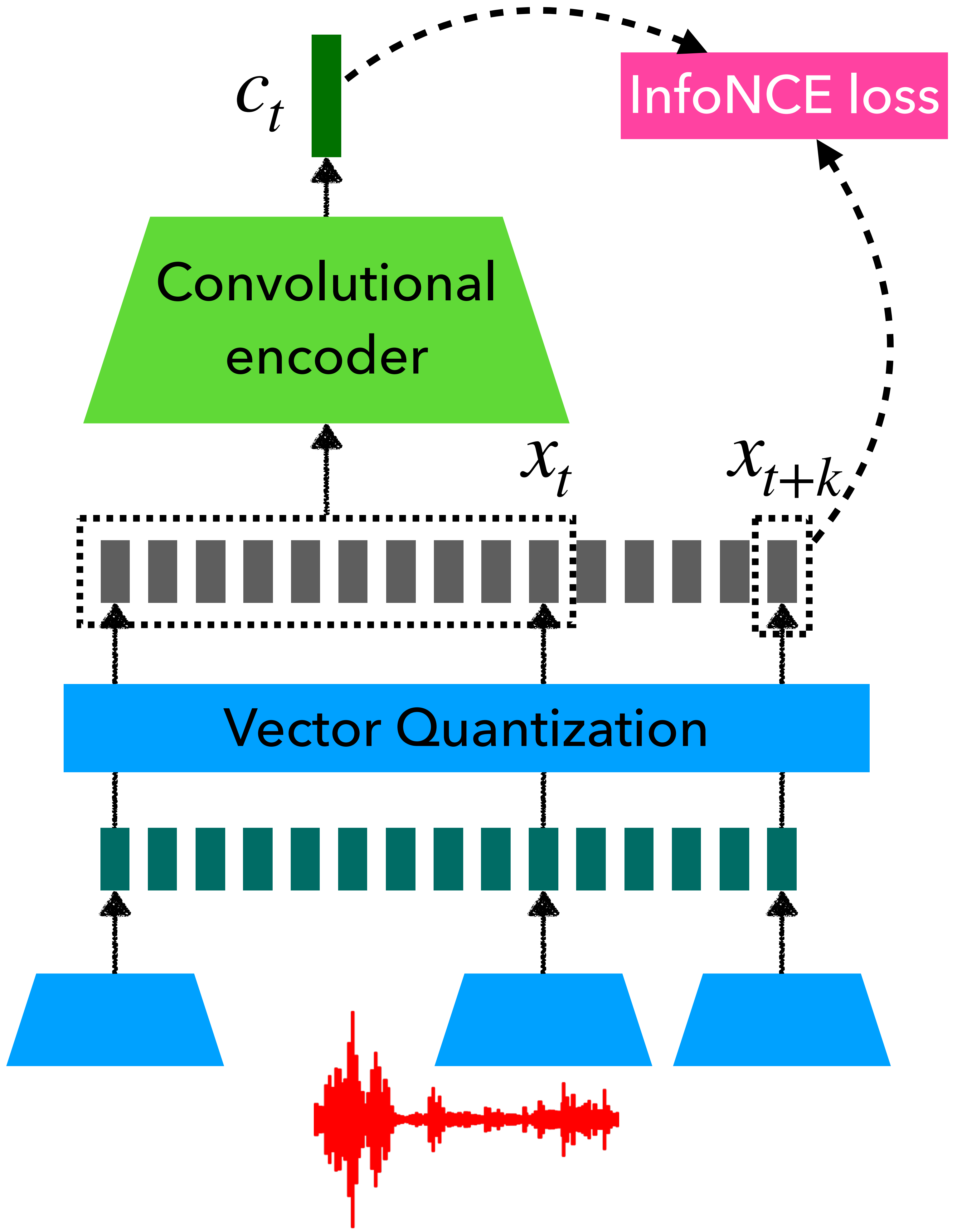
**Objective:** Predict vector-quantized feature vector k steps into the future using an autoregressive model

**Loss function:** L1-loss

# Prediction-based training (revisited)

## VQ-Wav2Vec

Baevski, Alexei et al. "vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations." *ArXiv* abs/1910.05453 (2020)




**Objective:** Predict future in vector-quantized *latent space* for next k steps using an autoregressive model

**Loss function:** InfoNCE loss

$$\mathcal{L}_N = -\,\mathbb{E}_X\left[\log \frac{f_k\left(x_{t+k}, c_t\right)}{\sum_{x_j \in X} f_k\left(x_j, c_t\right)}\right]$$

# Wav2vec 2.0

## Incorporate ideas from BERT

Baevski, Alexei et al. “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations.” *ArXiv* abs/2006.11477 (2020).

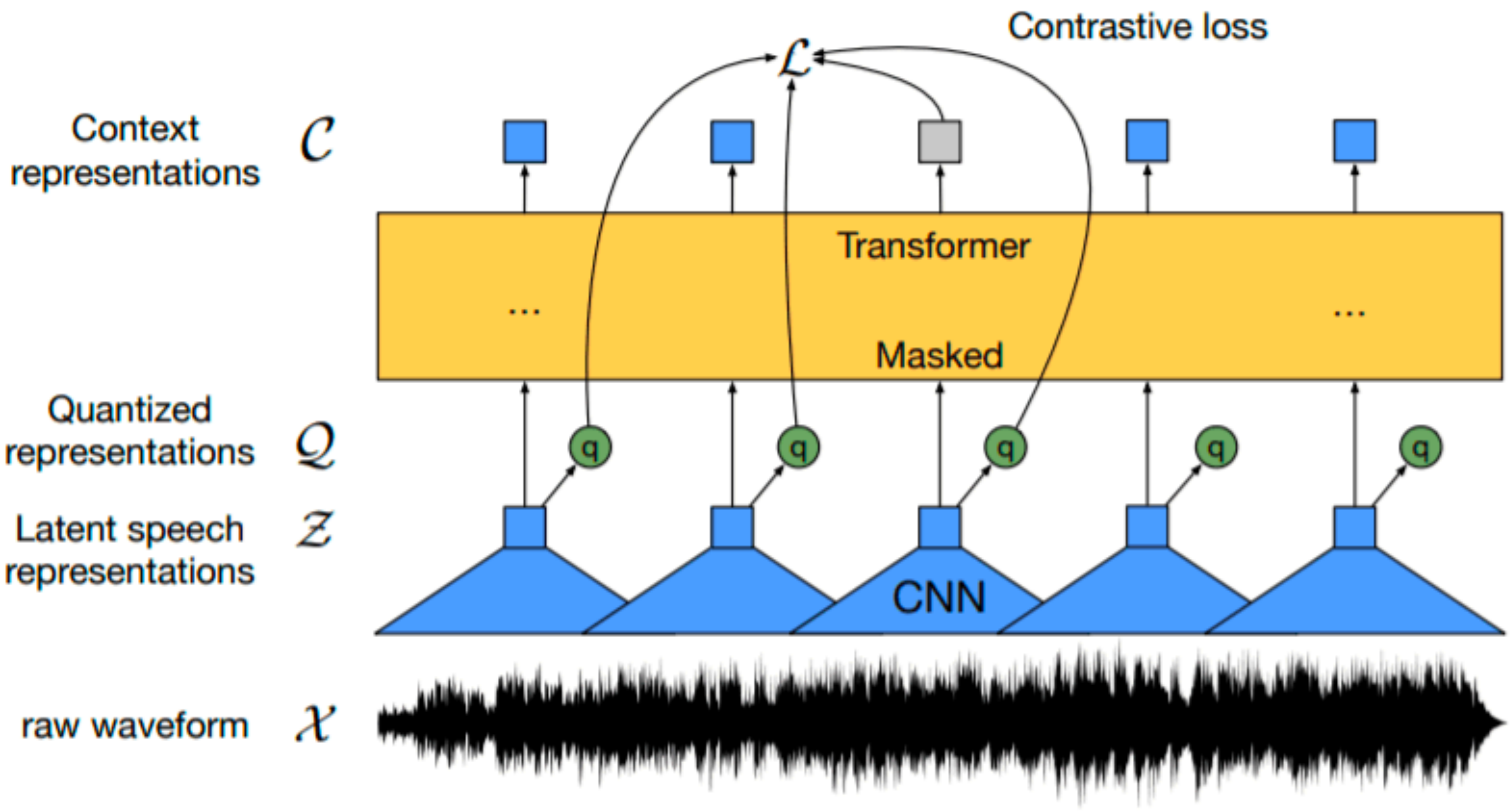



**Figure source:** Original paper from Baevski et al. (2020)

**Objective:** Predict **masked** vector-quantized representation using transformer

**Loss function:** InfoNCE loss

$$\mathscr{L}_N = -\mathbb{E}_X\left[\log \frac{f_k\left(x_{t+k}, c_t\right)}{\sum_{x_j \in X} f_k\left(x_j, c_t\right)}\right]$$

# Other pretraining methods

- Combine prediction and reconstruction losses (Wav2vec-C)
- Predict masked cluster index instead of quantized representation (HuBERT)
- Online teacher-student learning with mean-teacher method (SPIRAL)

# Summary of approaches