Supervised Approaches for Language and Speaker Recognition

PhD Thesis Defense Presentation
28th July 2023

Shreyas Ramoji

PhD Student,
Learning and Extraction of Acoustic Patterns (LEAP) Lab,
Electrical Engineering, Indian Institute of Science, Bangalore.

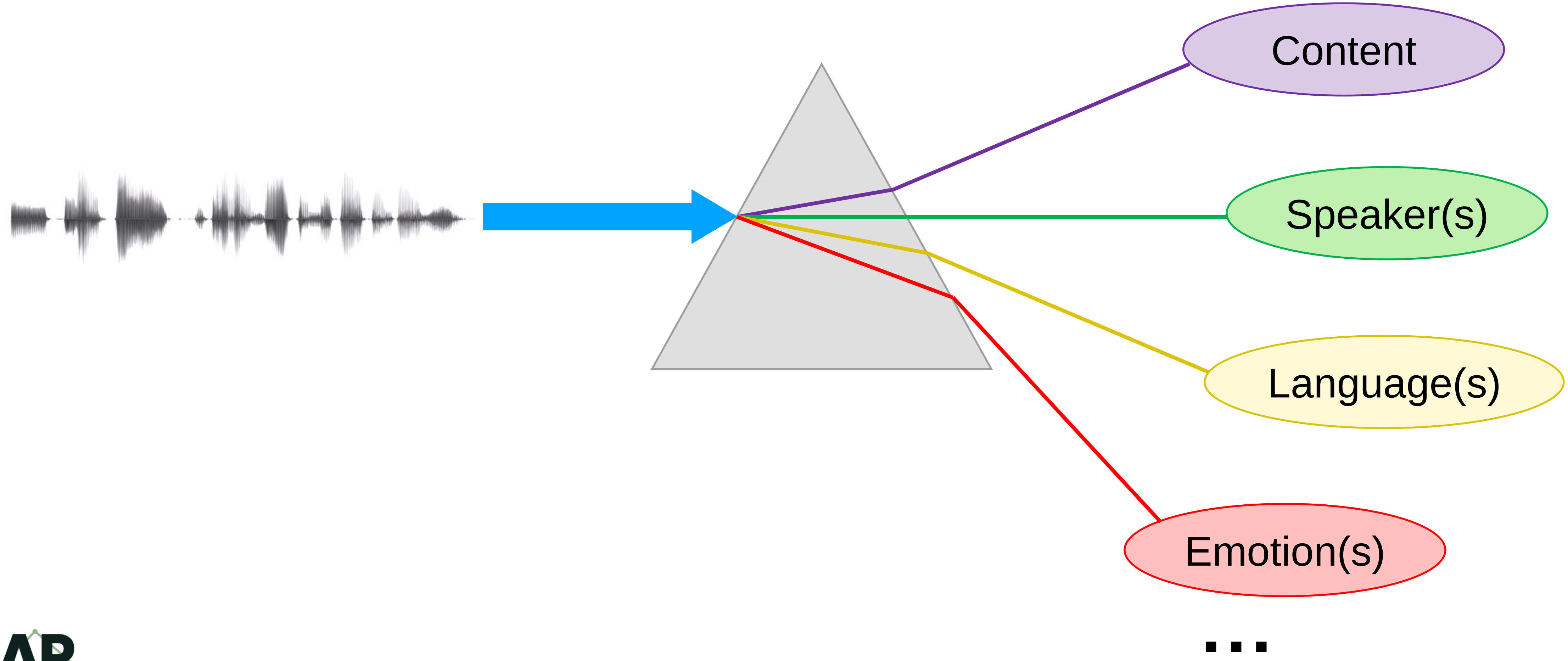
Advisor: Dr Sriram Ganapathy

Roadmap

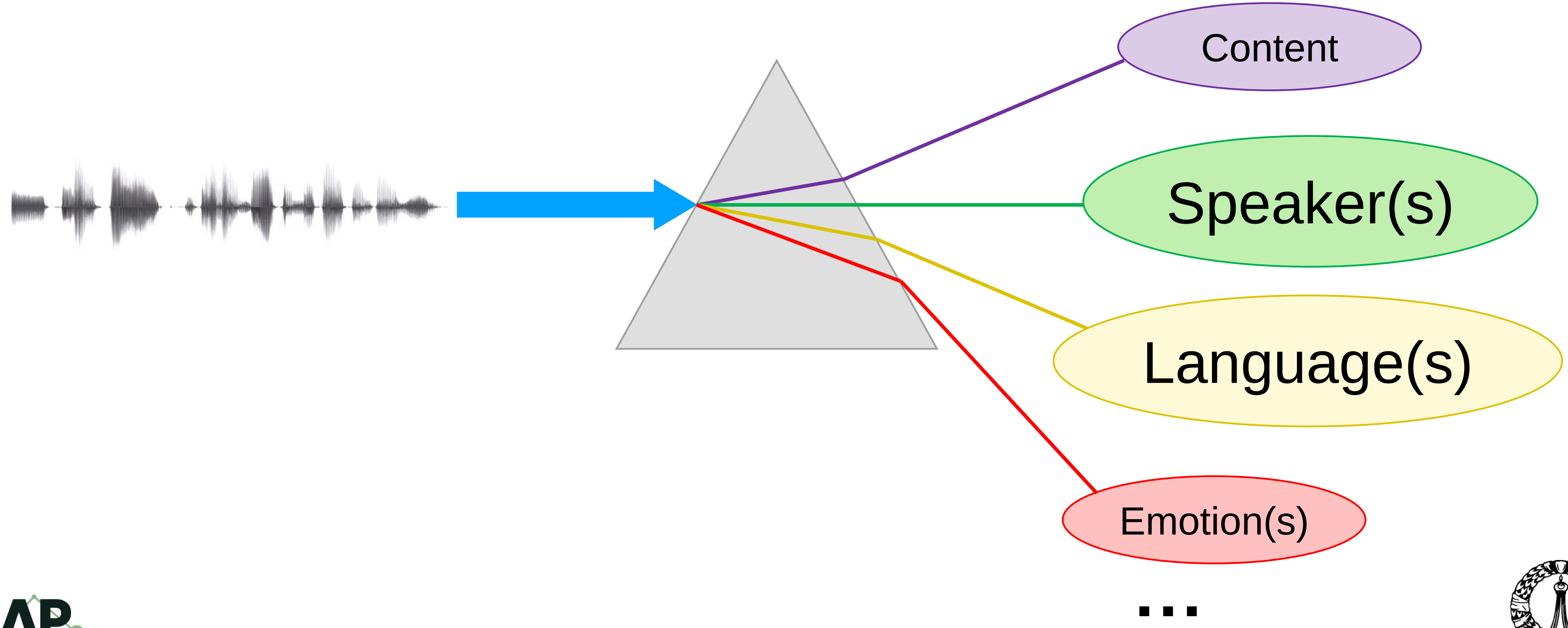
- Importance of Language and Speaker Recognition
- Brief Background Literature
- Supervised i-vector modeling for language recognition
- Supervised Neural Network Models for speaker verification
- Summary
- Recent advances and future perspectives

Introduction

Speech – A multitude of information

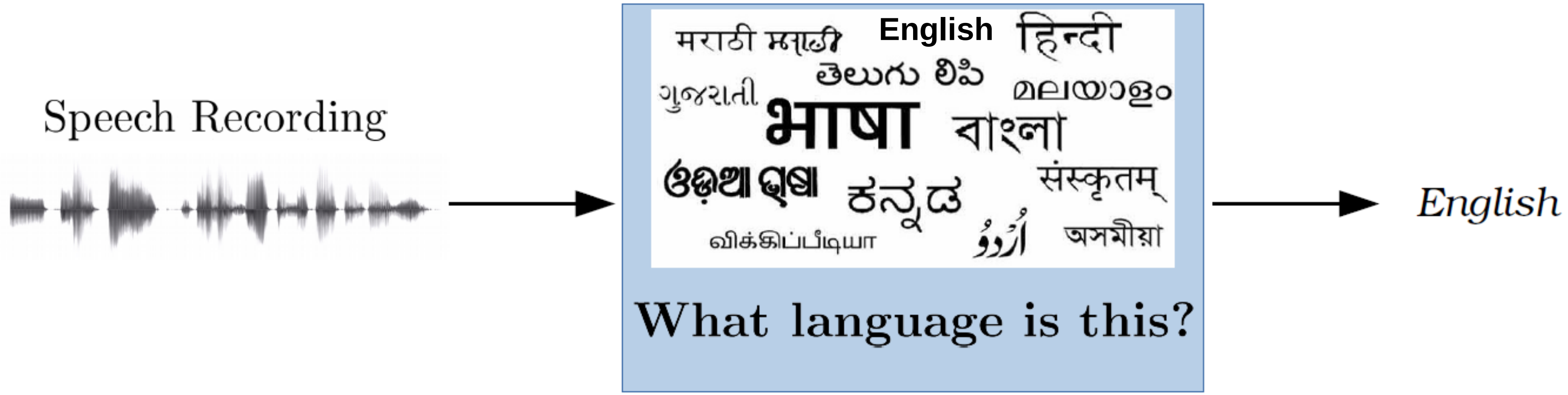


Speech – A multitude of information



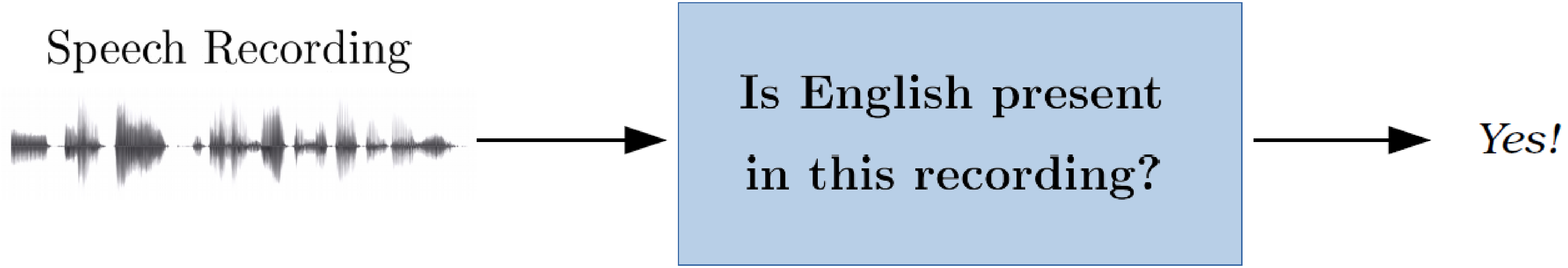
Problem statements

- Language/Accent Classification



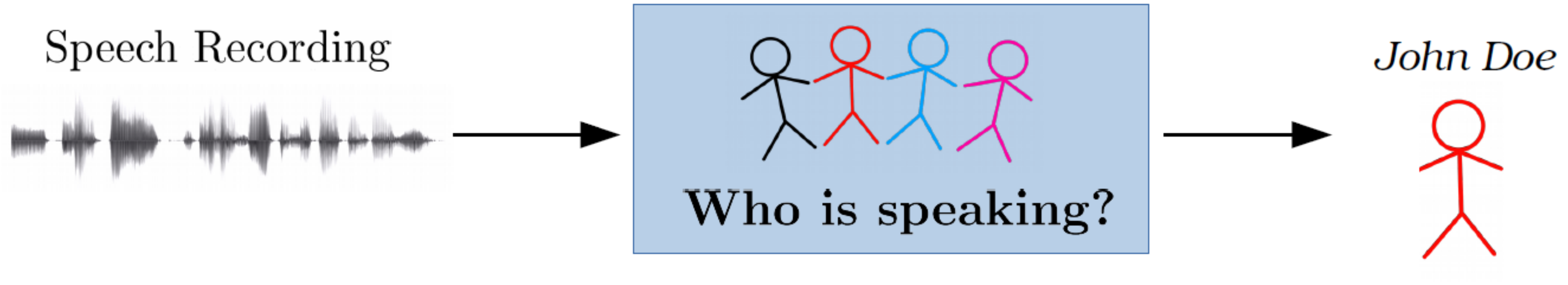
Problem statements

- Language/Accent Detection

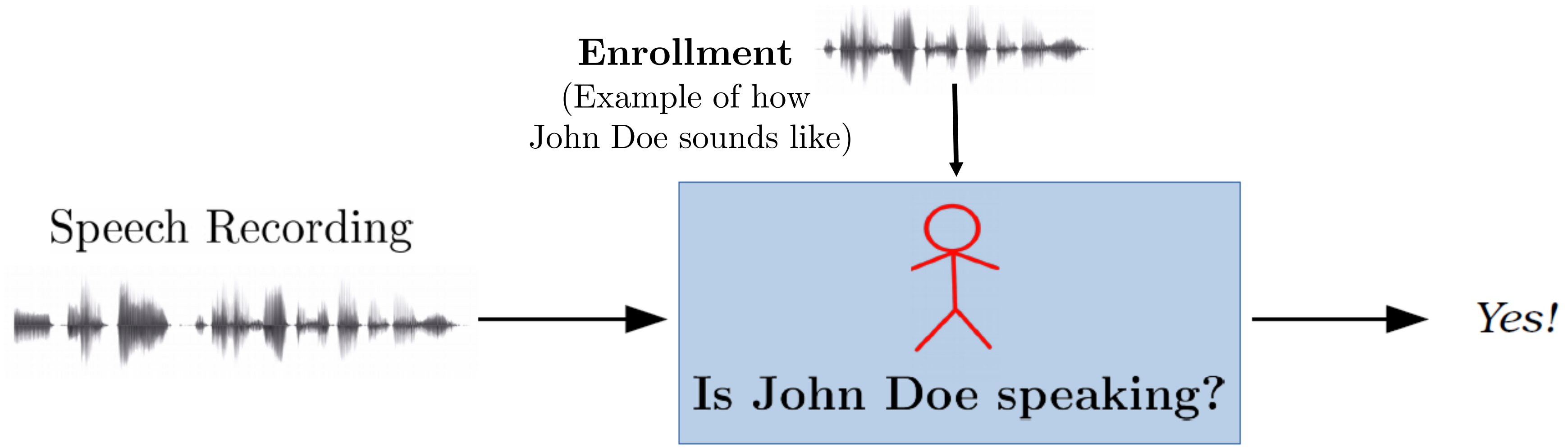


Problem statements

- Speaker Classification



- Speaker Verification / Detection



Applications

Applications

- Voice based authentication systems – In personal devices and call centers

Applications

- Voice based authentication systems – In personal devices and call centers
- Personalized smart home systems that responds to only authorized users

Applications

- Voice based authentication systems – In personal devices and call centers
- Personalized smart home systems that responds to only authorized users
- Enabling multi-lingual, accented speech technologies.

Applications

- Voice based authentication systems – In personal devices and call centers
- Personalized smart home systems that responds to only authorized users
- Enabling multi-lingual, accented speech technologies.
- Telephonic surveillance in defense applications

Applications

- Voice based authentication systems – In personal devices and call centers
- Personalized smart home systems that responds to only authorized users
- Enabling multi-lingual, accented speech technologies.
- Telephonic surveillance in defense applications
- Criminal investigations

Applications

- Voice based authentication systems – In personal devices and call centers
- Personalized smart home systems that responds to only authorized users
- Enabling multi-lingual, accented speech technologies.
- Telephonic surveillance in defense applications
- Criminal investigations
- Basis for Speaker & Language Diarization & Multi-speaker /Multilingual speech technologies.

What makes these problems challenging?

What makes these problems challenging?

- Short duration of target speaker enrollment data / short test duration / both

What makes these problems challenging?

- Short duration of target speaker enrollment data / short test duration / both

- Noise, reverberation, low SNR conditions

What makes these problems challenging?

- Short duration of target speaker enrollment data / short test duration / both
- Noise, reverberation, low SNR conditions
- Recording environment and Microphone quality differences between enrollment and test recordings

What makes these problems challenging?

- Short duration of target speaker enrollment data / short test duration / both
- Noise, reverberation, low SNR conditions
- Recording environment and Microphone quality differences between enrollment and test recordings
- Language & accent variability / Speaker variability / Content Variability

What makes these problems challenging?

- Short duration of target speaker enrollment data / short test duration / both
- Noise, reverberation, low SNR conditions
- Recording environment and Microphone quality differences between enrollment and test recordings
- Language & accent variability / Speaker variability / Content Variability
- Variations in emotion and prosody

What makes these problems challenging?

- Short duration of target speaker enrollment data / short test duration / both
- Noise, reverberation, low SNR conditions
- Recording environment and Microphone quality differences between enrollment and test recordings
- Language & accent variability / Speaker variability / Content Variability
- Variations in emotion and prosody
- Mimicry and spoofing

Why Language & Speaker Recognition?

Why Language & Speaker Recognition?

- Speaker and language are both segment level properties.
    - Segments of duration ranging from as low as 1 second to a few minutes can be associated with a single speaker/language label.
    - A certain duration of speech must be processed to determine the speaker/language.
    - Longer the duration, lesser is the uncertainty of speaker and language.

Why Language & Speaker Recognition?

- Speaker and language are both segment level properties.
    - Segments of duration ranging from as low as 1 second to a few minutes can be associated with a single speaker/language label.
    - A certain duration of speech must be processed to determine the speaker/language.
    - Longer the duration, lesser is the uncertainty of speaker and language.
- A common practice in Speaker/Language Recognition research is to use the approaches used in speaker recognition for language recognition, and vice-versa.

Why Language & Speaker Recognition?

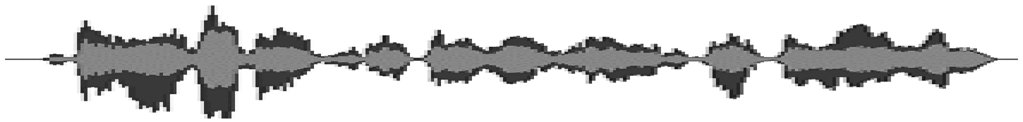
- Speaker and language are both segment level properties.
  - Segments of duration ranging from as low as 1 second to a few minutes can be associated with a single speaker/language label.
  - A certain duration of speech must be processed to determine the speaker/language.
  - Longer the duration, lesser is the uncertainty of speaker and language.
- A common practice in Speaker/Language Recognition research is to use the approaches used in speaker recognition for language recognition, and vice-versa.
- Similar approaches are found to achieve the state-of-the-art in both speaker and language recognition.

A Brief Background…

Gaussian Mixture Models

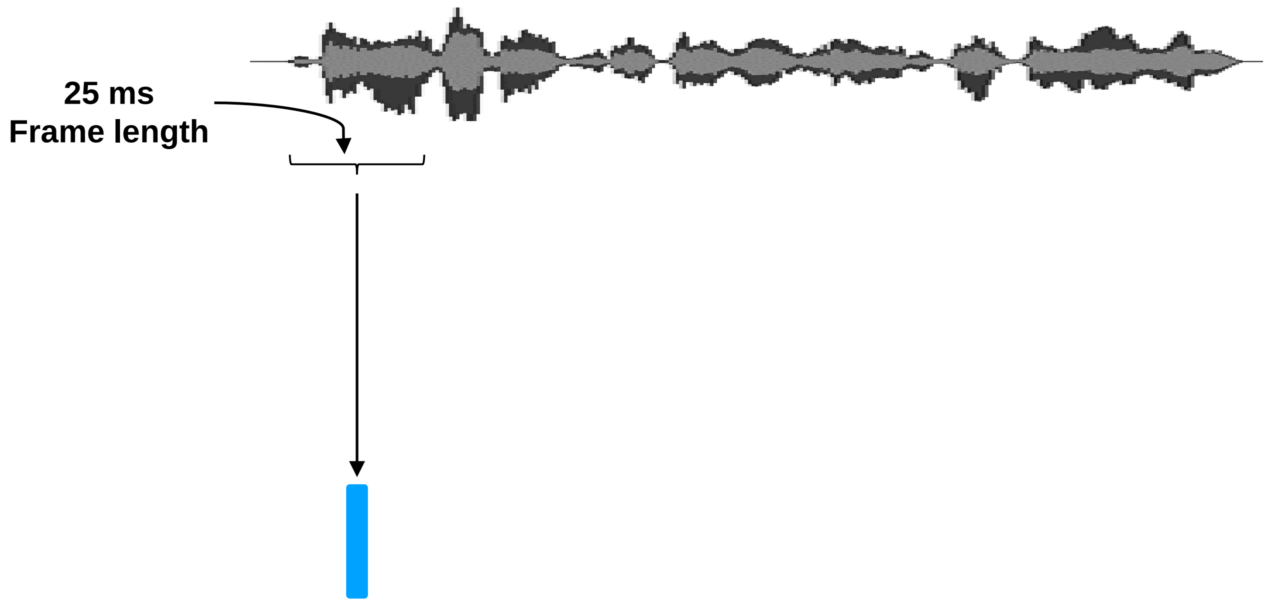
Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

Gaussian Mixture Models



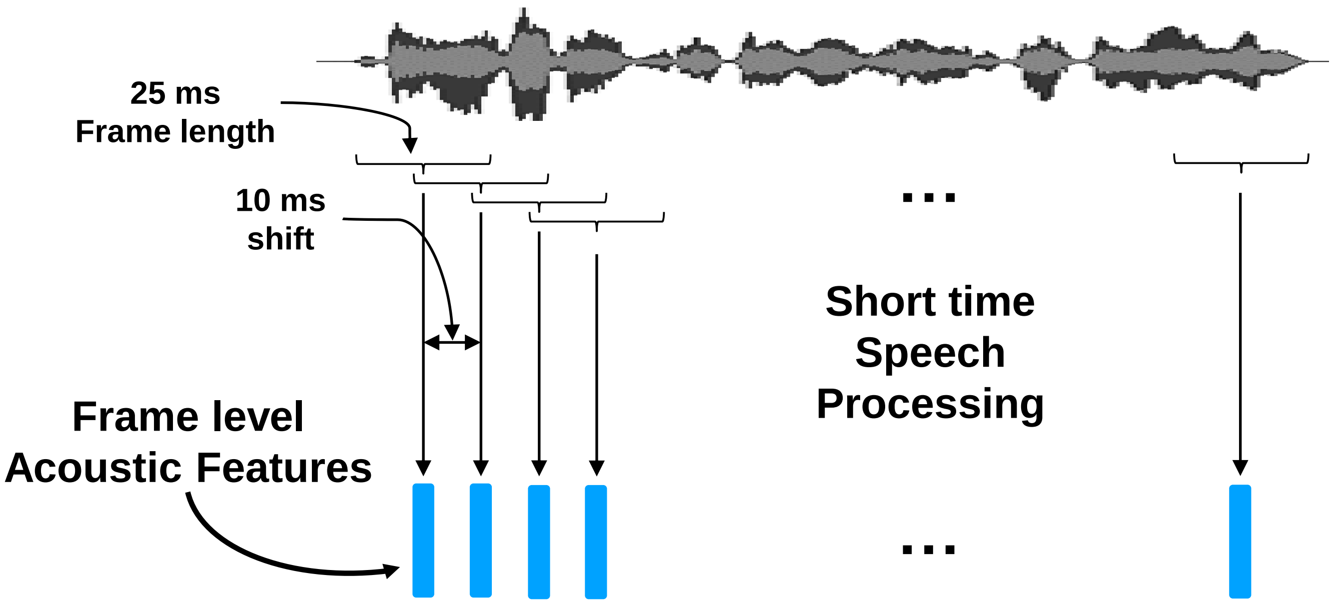
Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

Gaussian Mixture Models



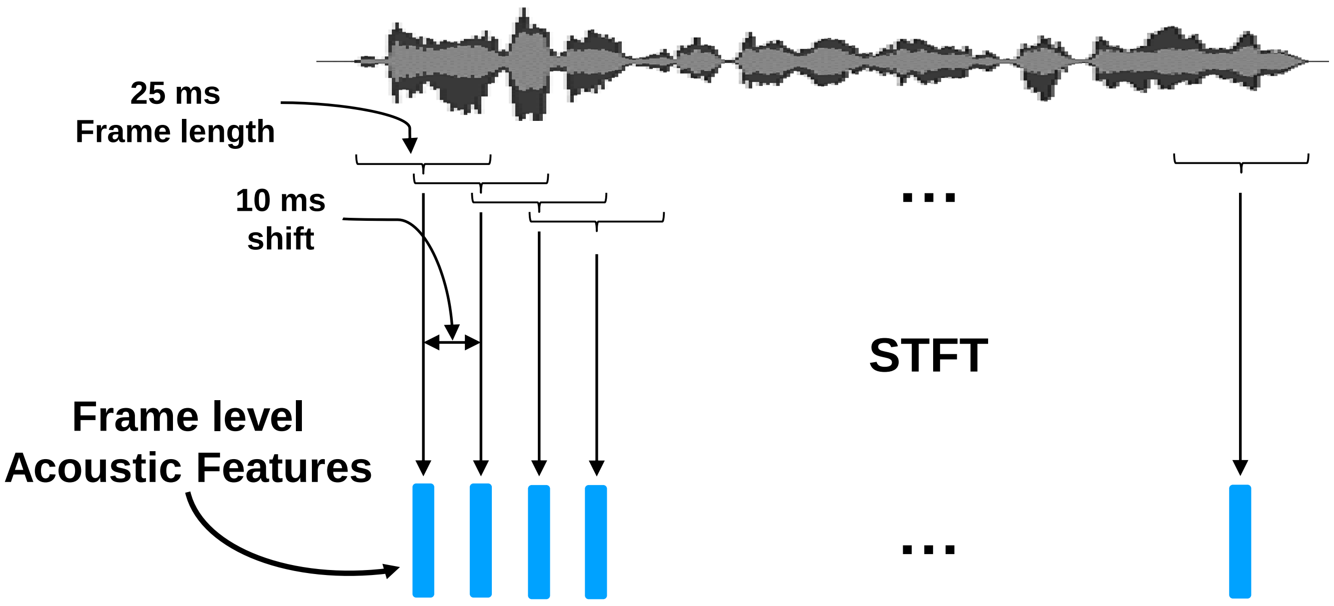
Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

Gaussian Mixture Models



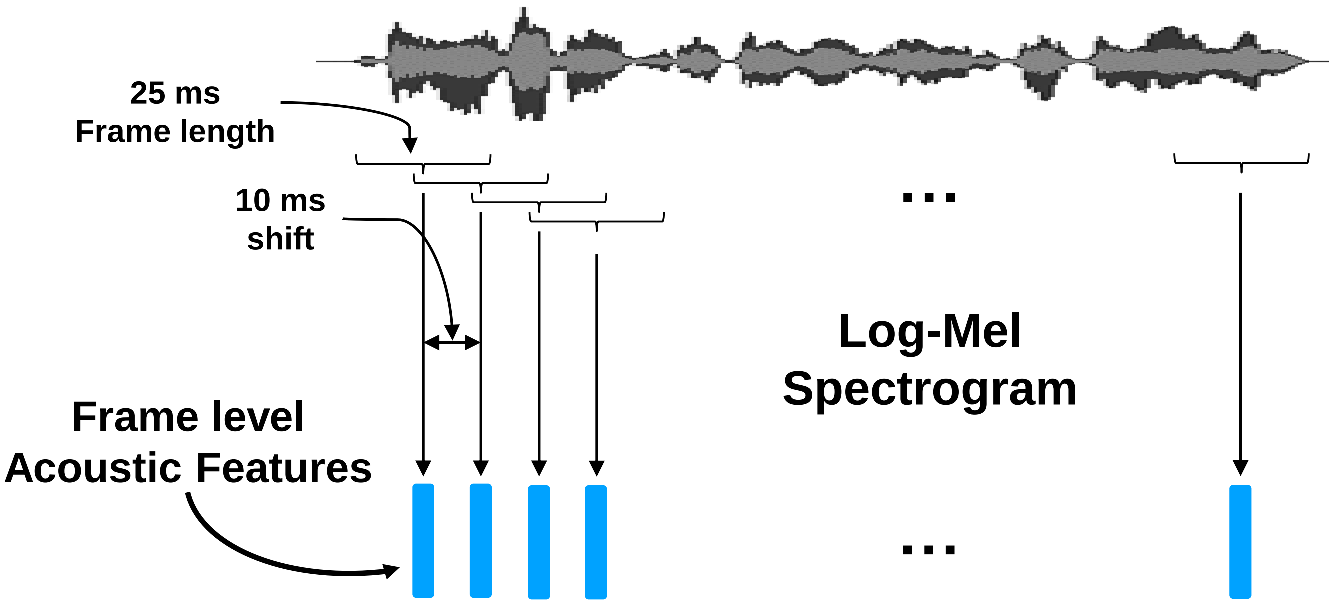
Douglas Reynolds, “*A Gaussian mixture modeling approach to text-independent speaker identification*”, 1992.

Gaussian Mixture Models



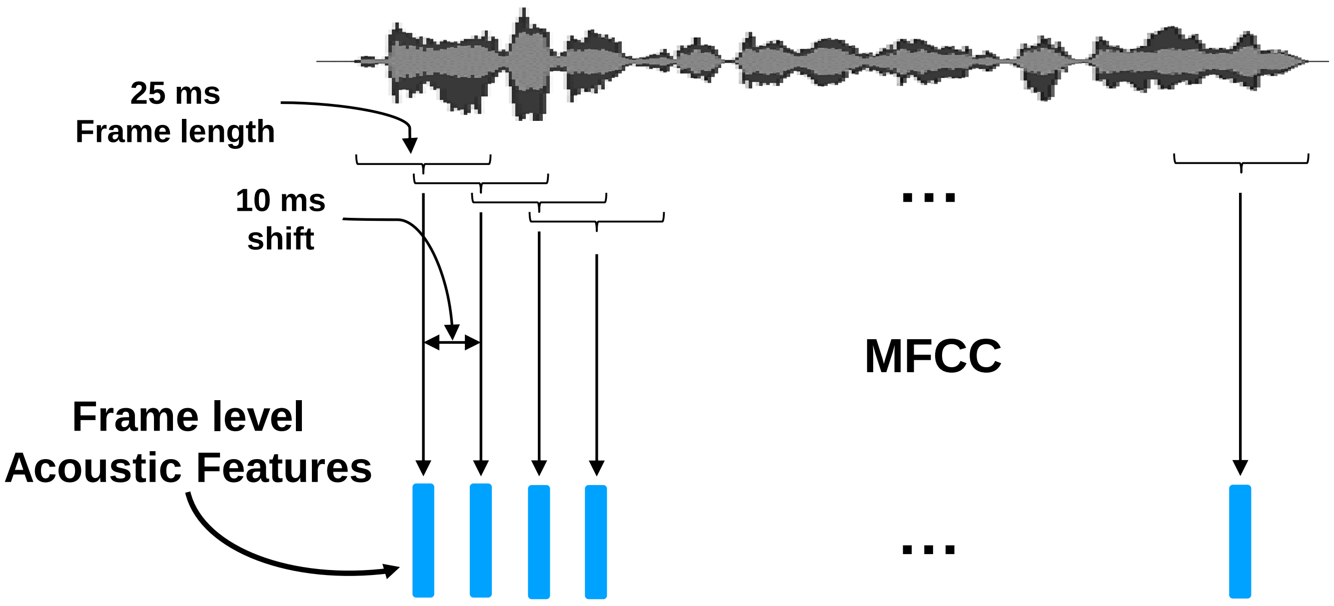
Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

Gaussian Mixture Models



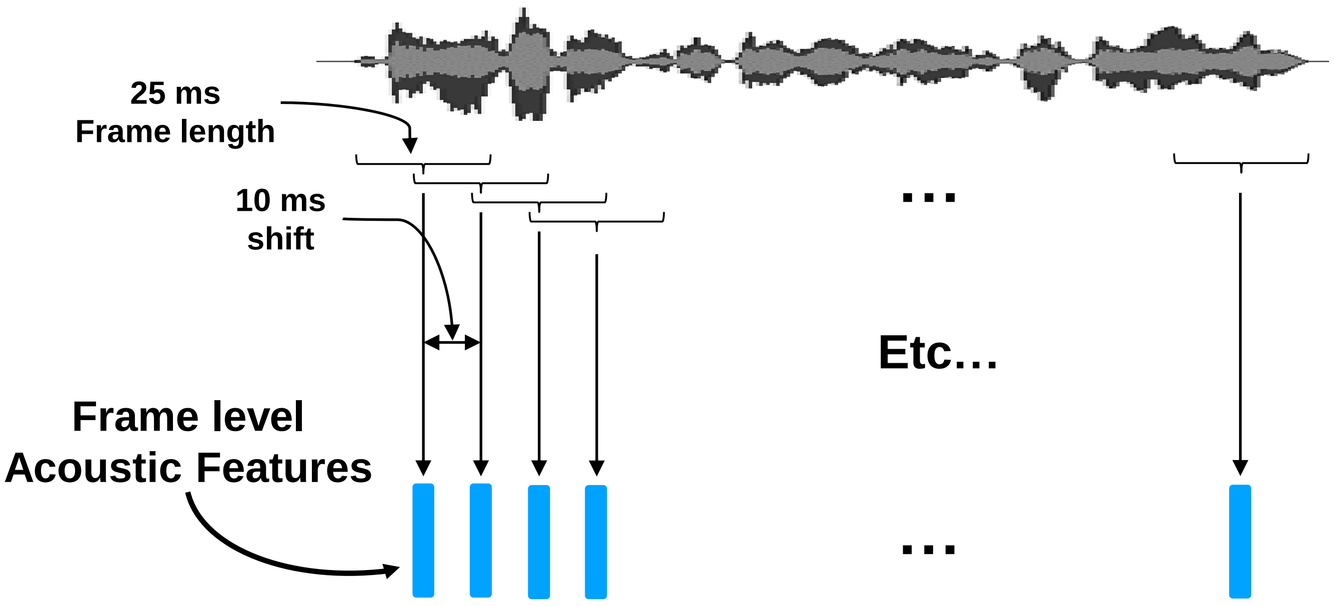
Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

Gaussian Mixture Models



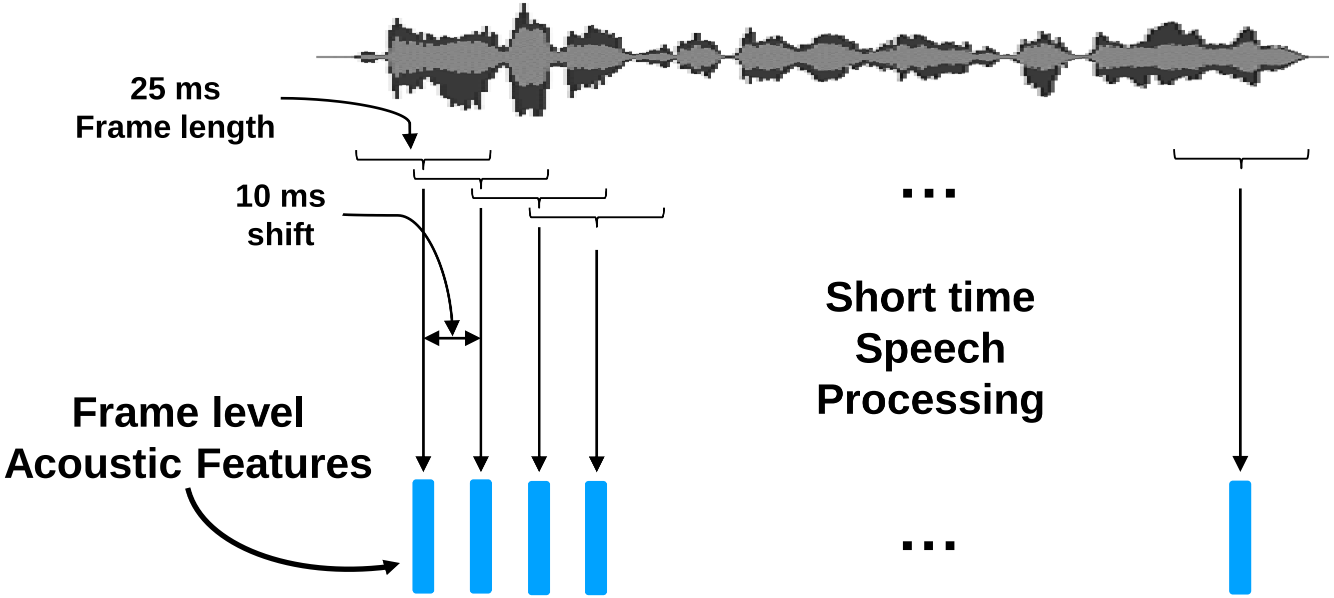
Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

Gaussian Mixture Models



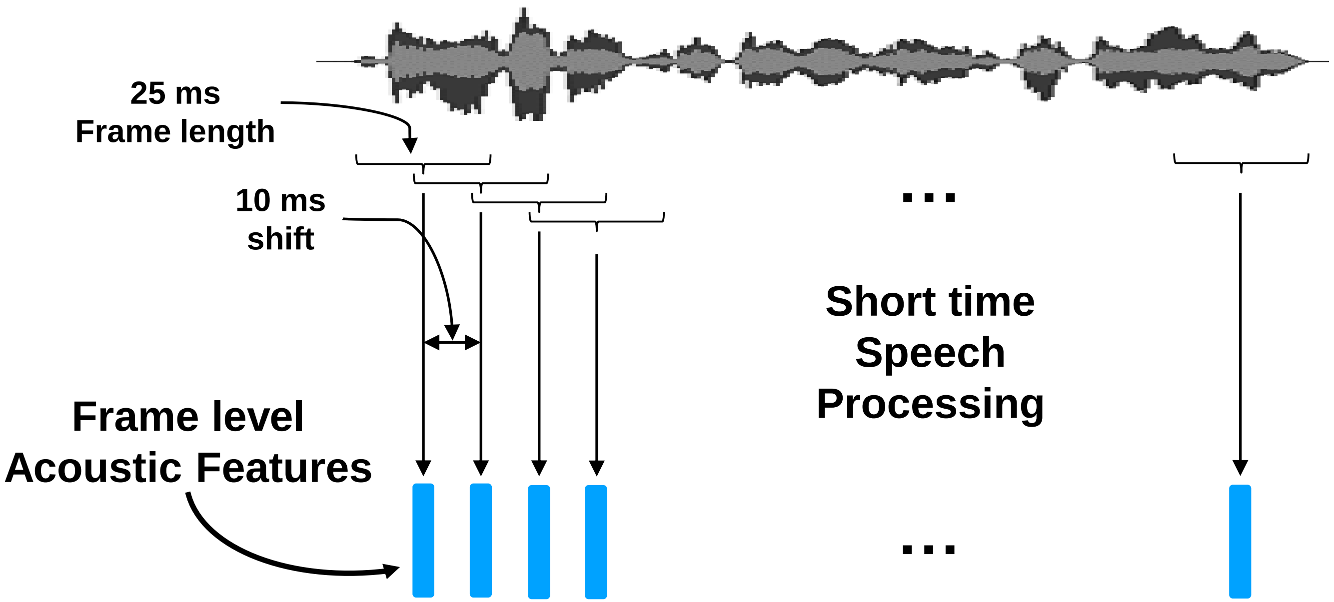
Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

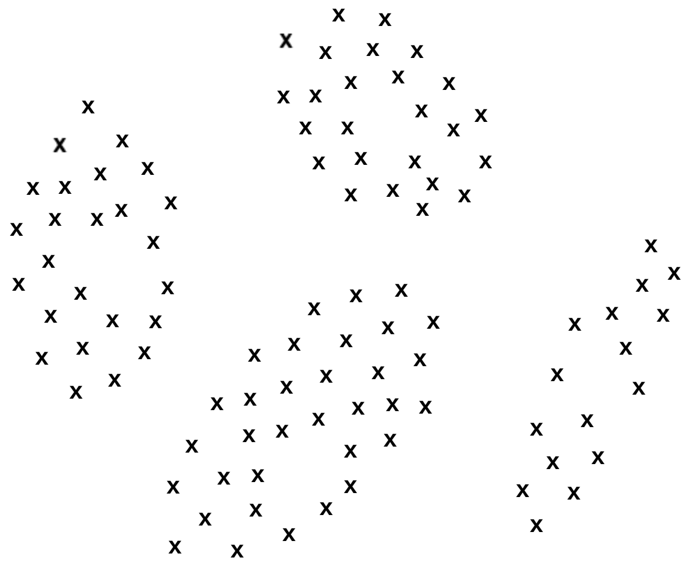
Gaussian Mixture Models



Douglas Reynolds, “*A Gaussian mixture modeling approach to text-independent speaker identification*”, 1992.

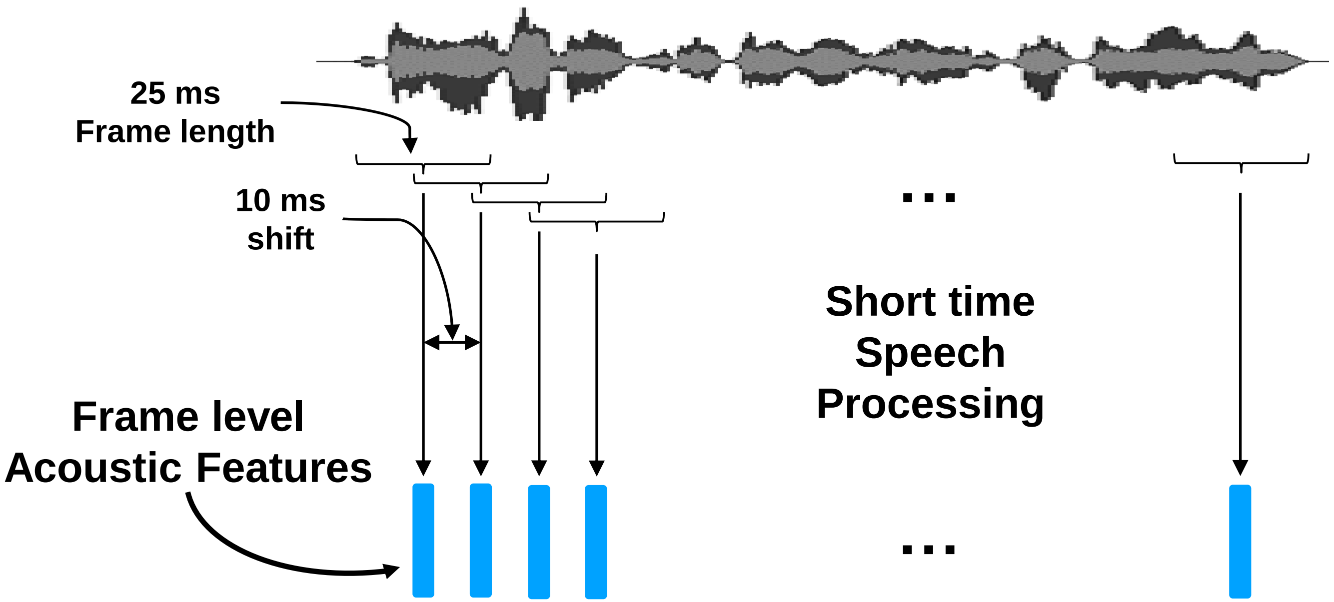
Gaussian Mixture Models

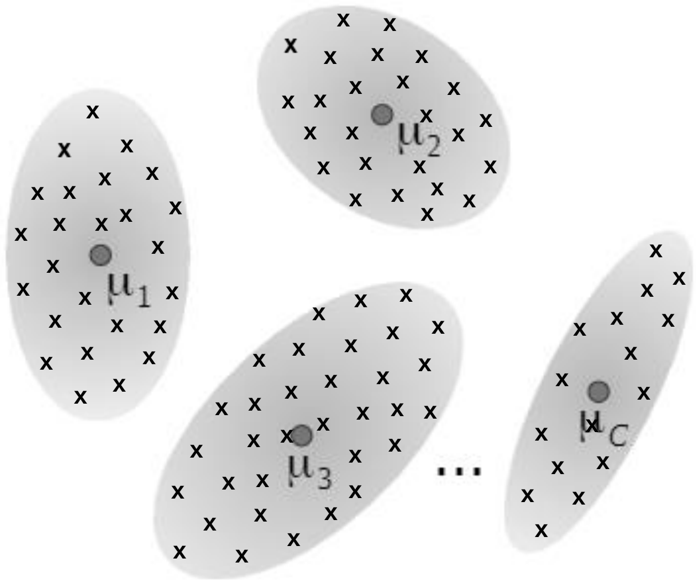




Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

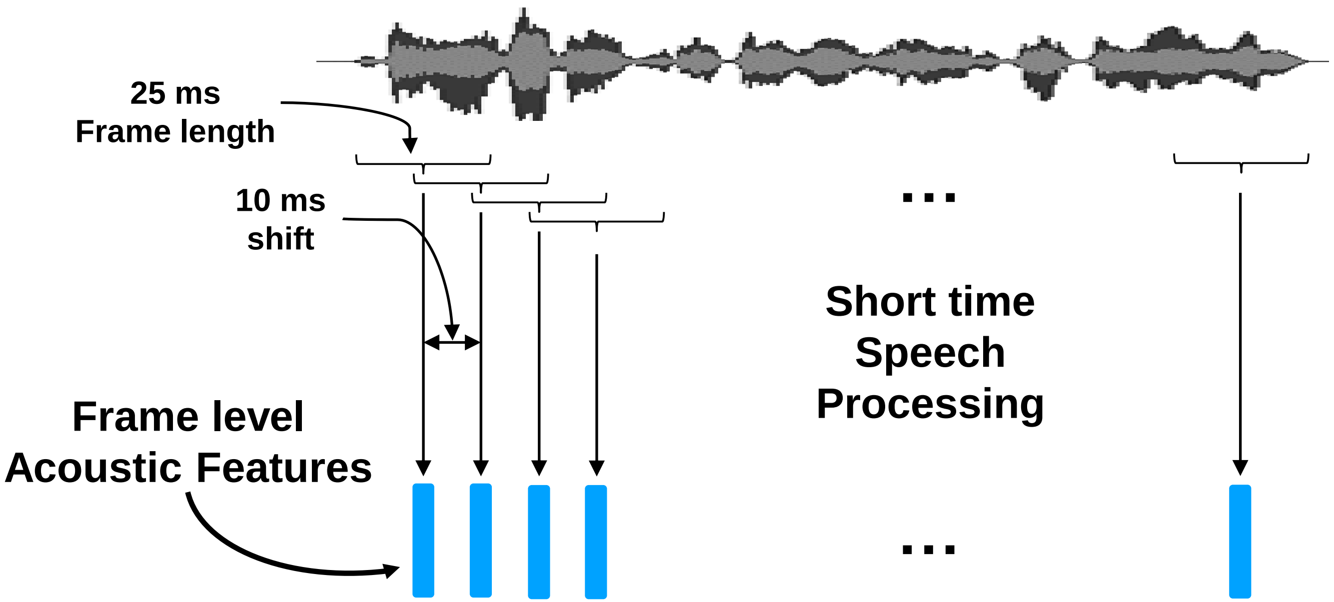
Gaussian Mixture Models

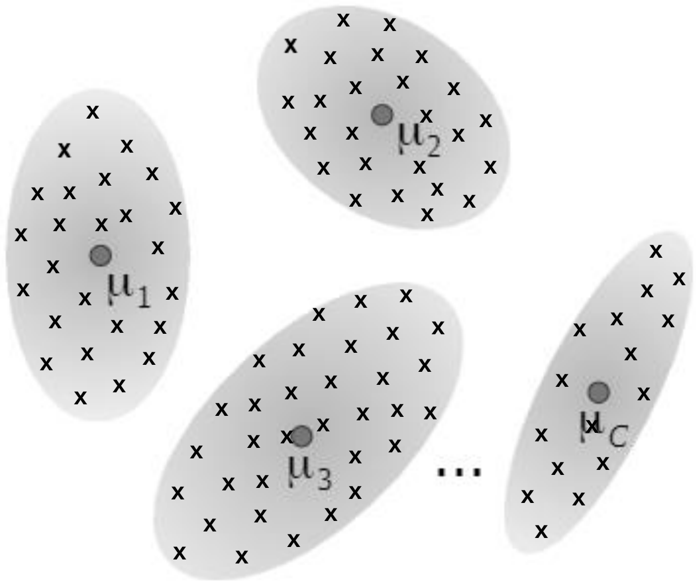


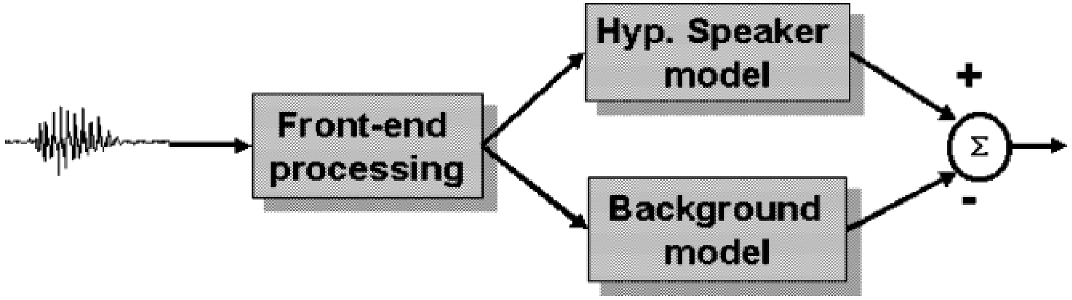


Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

Gaussian Mixture Models







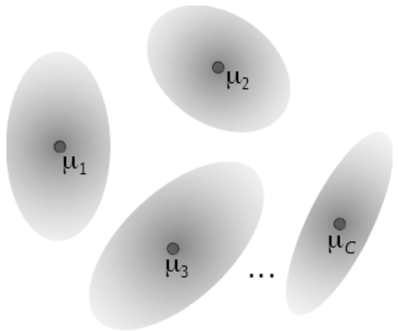
$$\frac{p(Y \mid H_0)}{p(Y \mid H_1)} \begin{cases} \geq \theta & \text{accept } H_0 \\ < \theta & \text{reject } H_0, \end{cases}$$

Douglas Reynolds, "*A Gaussian mixture modeling approach to text-independent speaker identification*", 1992.

Adaptation of GMM

Adaptation of GMM

- A large dataset of speech, with many speakers is used to estimate a GMM Background Model.



GMM-UBM

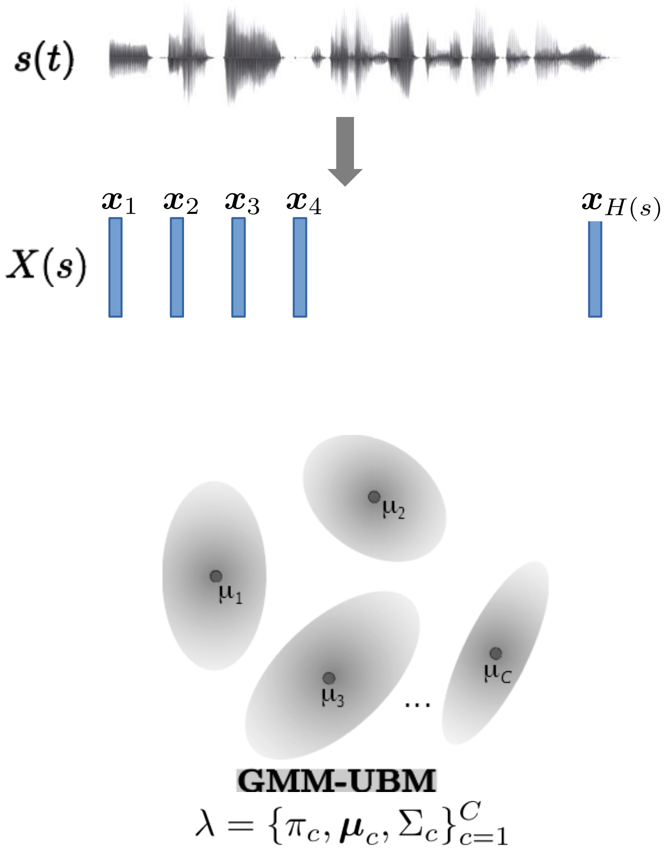
$$\lambda = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^{C}$$

Douglas Reynolds et. al., "*Speaker Verification Using Adapted Gaussian Mixture Models*", 2000.

Adaptation of GMM

- A large dataset of speech, with many speakers is used to estimate a GMM Background Model.

- The enrollment recordings of the target speaker or language are used to adapt the parameters of the Background model to get the target speaker / language model.

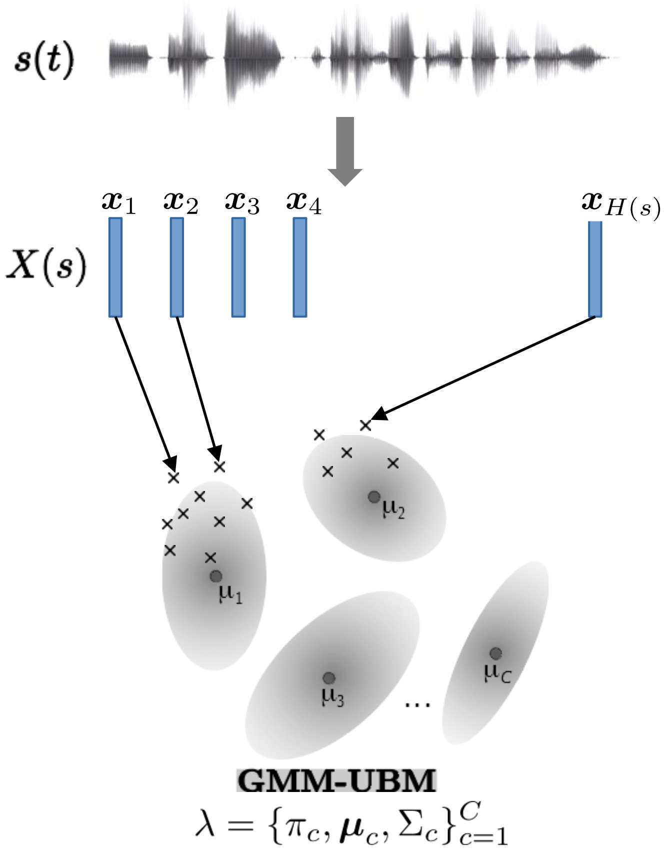


Douglas Reynolds et. al., “*Speaker Verification Using Adapted Gaussian Mixture Models*”, 2000.

Adaptation of GMM

- A large dataset of speech, with many speakers is used to estimate a GMM Background Model.

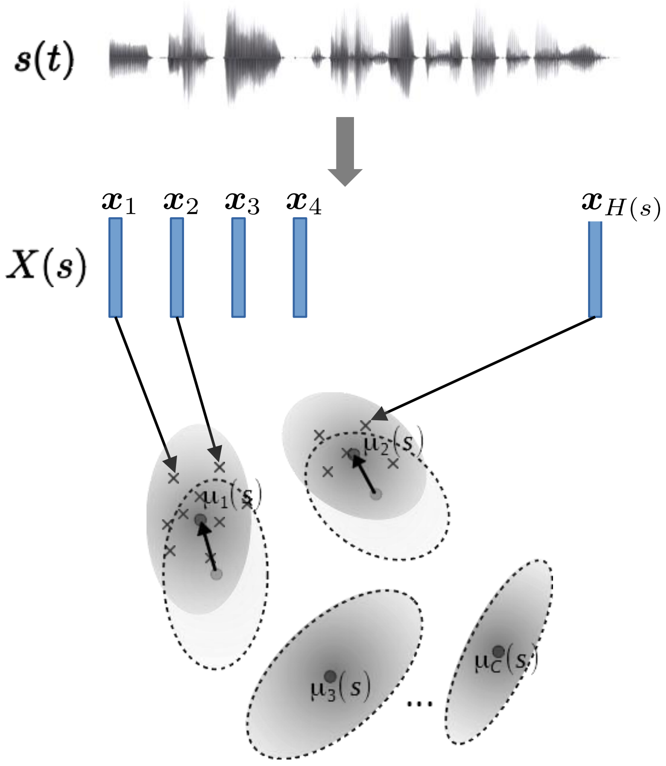
- The enrollment recordings of the target speaker or language are used to adapt the parameters of the Background model to get the target speaker / language model.



Douglas Reynolds et. al., "*Speaker Verification Using Adapted Gaussian Mixture Models*", 2000.

Adaptation of GMM

- A large dataset of speech, with many speakers is used to estimate a GMM Background Model.
- The enrollment recordings of the target speaker or language are used to adapt the parameters of the Background model to get the target speaker / language model.

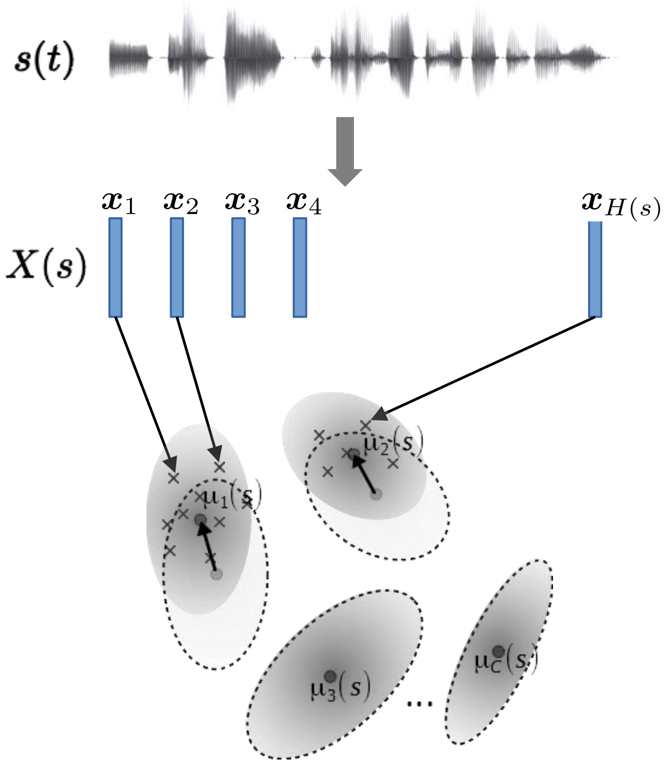


$$\lambda(s) = \{\pi_c, \boldsymbol{\mu}_c(s), \Sigma_c\}_{c=1}^{C}$$

Douglas Reynolds et. al., "*Speaker Verification Using Adapted Gaussian Mixture Models*", 2000.

Adaptation of GMM

- A large dataset of speech, with many speakers is used to estimate a GMM Background Model.
- The enrollment recordings of the target speaker or language are used to adapt the parameters of the Background model to get the target speaker / language model.
- The GMM log likelihoods of a test utterance is computed using the Background model (null hypothesis) and target speaker model (target hypothesis)… and then we compute a log likelihood ratio.
- Decision is made by thresholding the LLR based on the required criterion (Min Bayes Risk, Neyman-Pearson)



$$\lambda(s) = \{\pi_c, \boldsymbol{\mu}_c(s), \Sigma_c\}_{c=1}^{C}$$

Douglas Reynolds et. al., "*Speaker Verification Using Adapted Gaussian Mixture Models*", 2000.

Joint Factor Analysis

Kenny, “*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*”, 2006

Joint Factor Analysis

Kenny, “*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*”, 2006

Joint Factor Analysis

$$\boldsymbol{M}(\boldsymbol{s}) = \begin{pmatrix} \boldsymbol{\mu_1}(\boldsymbol{s}) \\ \boldsymbol{\mu_2}(\boldsymbol{s}) \\ \vdots \\ \boldsymbol{\mu_C}(\boldsymbol{s}) \end{pmatrix}_{\boldsymbol{CF \times 1}}$$

Kenny, "*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*", 2006

Joint Factor Analysis

$$\boldsymbol{M}(\boldsymbol{s}) = \begin{pmatrix} \boldsymbol{\mu}_1(\boldsymbol{s}) \\ \boldsymbol{\mu}_2(\boldsymbol{s}) \\ \vdots \\ \boldsymbol{\mu}_C(\boldsymbol{s}) \end{pmatrix}_{CF\times 1}$$

Kenny, "*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*", 2006

Joint Factor Analysis

$$\boldsymbol{M(s)} = \begin{pmatrix} \boldsymbol{\mu_1(s)} \\ \boldsymbol{\mu_2(s)} \\ \vdots \\ \boldsymbol{\mu_C(s)} \end{pmatrix}_{\boldsymbol{CF\times 1}}$$

$$\boldsymbol{M(s)} = \mathbf{M} + \mathbf{Vy} + \mathbf{Ux} + \mathbf{Dz}$$

Kenny, "*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*", 2006

Joint Factor Analysis

$$\boldsymbol{M}(\boldsymbol{s}) = \begin{pmatrix} \boldsymbol{\mu}_1(\boldsymbol{s}) \\ \boldsymbol{\mu}_2(\boldsymbol{s}) \\ \vdots \\ \boldsymbol{\mu}_C(\boldsymbol{s}) \end{pmatrix}_{\boldsymbol{CF \times 1}}$$

$$\boldsymbol{M}(\boldsymbol{s}) = \mathbf{M} + \mathbf{Vy} + \mathbf{Ux} + \mathbf{Dz}$$

supervector

Kenny, "*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*", 2006

Joint Factor Analysis

$$\boldsymbol{M}(\boldsymbol{s}) = \begin{pmatrix} \boldsymbol{\mu}_1(\boldsymbol{s}) \\ \boldsymbol{\mu}_2(\boldsymbol{s}) \\ \vdots \\ \boldsymbol{\mu}_C(\boldsymbol{s}) \end{pmatrix}_{\boldsymbol{CF \times 1}}$$

$$\boldsymbol{M}(\boldsymbol{s}) = \mathbf{M} + \mathbf{Vy} + \mathbf{Ux} + \mathbf{Dz}$$

supervector

UBM Mean component

LEAP

Kenny, "*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*", 2006

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Joint Factor Analysis

$$M(s) = \begin{pmatrix} \mu_1(s) \\ \mu_2(s) \\ \vdots \\ \mu_C(s) \end{pmatrix}_{CF\times 1}$$

$$\underbrace{M(s)}_{\text{supervector}} = \underbrace{\mathbf{M}}_{\text{UBM Mean component}} + \underbrace{\mathbf{Vy}}_{\text{Speaker dependent component}} + \mathbf{Ux} + \mathbf{Dz}$$

Kenny, "*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*", 2006

Joint Factor Analysis

$$M(s) = \begin{pmatrix} \mu_1(s) \\ \mu_2(s) \\ \vdots \\ \mu_C(s) \end{pmatrix}_{CF\times 1}$$

$$\underbrace{M(s)}_{\text{supervector}} = \underbrace{\mathbf{M}}_{\text{UBM Mean component}} + \underbrace{\mathbf{Vy}}_{\text{Speaker dependent component}} + \underbrace{\mathbf{Ux}}_{\text{Channel Dependent component}} + \mathbf{Dz}$$

Kenny, "*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*", 2006

Joint Factor Analysis

$$\boldsymbol{M}(\boldsymbol{s}) = \begin{pmatrix} \mu_1(s) \\ \mu_2(s) \\ \vdots \\ \mu_C(s) \end{pmatrix}_{CF\times 1}$$

$$\underbrace{\boldsymbol{M}(\boldsymbol{s})}_{\text{supervector}} = \underbrace{\mathbf{M}}_{\text{UBM Mean component}} + \underbrace{\mathbf{Vy}}_{\text{Speaker dependent component}} + \underbrace{\mathbf{Ux}}_{\text{Channel Dependent component}} + \underbrace{\mathbf{Dz}}_{\text{Residual component}}$$

Kenny, "*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*", 2006

Joint Factor Analysis

$$M(s) = \begin{pmatrix} \mu_1(s) \\ \mu_2(s) \\ \vdots \\ \mu_C(s) \end{pmatrix}_{CF\times 1}$$

$$M(s) = \mathbf{M} + \mathbf{Vy} + \mathbf{Ux} + \mathbf{Dz}$$

supervector — $M(s)$

UBM Mean component — $\mathbf{M}$

Speaker dependent component — $\mathbf{Vy}$

Channel Dependent component — $\mathbf{Ux}$

Residual component — $\mathbf{Dz}$

Where: $\mathbf{x}, \mathbf{y}, \mathbf{z} \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$

Kenny, "*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*", 2006

Joint Factor Analysis

- The Adapted GMM mean components concatenated as a single vector is called a “supervector”.

$$\boldsymbol{M}(\boldsymbol{s}) = \begin{pmatrix} \boldsymbol{\mu_1}(\boldsymbol{s}) \\ \boldsymbol{\mu_2}(\boldsymbol{s}) \\ \vdots \\ \boldsymbol{\mu_C}(\boldsymbol{s}) \end{pmatrix}_{\boldsymbol{CF\times 1}}$$

- The supervector of an utterance $\boldsymbol{s}$ is modelled as:

$$\boldsymbol{M}(\boldsymbol{s}) = \mathbf{M} + \mathbf{Vy} + \mathbf{Ux} + \mathbf{Dz}$$

supervector

UBM Mean component

Speaker dependent component

Channel Dependent component

Residual component

Where: $\mathbf{x}, \mathbf{y}, \mathbf{z} \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$

Kenny, “*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*”, 2006

Joint Factor Analysis

- The Adapted GMM mean components concatenated as a single vector is called a “supervector”.

$$\boldsymbol{M}(\boldsymbol{s}) = \begin{pmatrix} \boldsymbol{\mu}_1(\boldsymbol{s}) \\ \boldsymbol{\mu}_2(\boldsymbol{s}) \\ \vdots \\ \boldsymbol{\mu}_C(\boldsymbol{s}) \end{pmatrix}_{\boldsymbol{CF}\times\mathbf{1}}$$

- The supervector of an utterance $\boldsymbol{s}$ is modelled as:

$$\underbrace{\boldsymbol{M}(\boldsymbol{s})}_{\text{supervector}} = \underbrace{\mathbf{M}}_{\text{UBM Mean component}} + \underbrace{\mathbf{Vy}}_{\text{Speaker dependent component}} + \underbrace{\mathbf{Ux}}_{\text{Channel Dependent component}} + \underbrace{\mathbf{Dz}}_{\text{Residual component}}$$

Where: $\mathbf{x}, \mathbf{y}, \mathbf{z} \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$

$$\mathbf{LLR} = \mathbf{log}\left(\frac{\mathbf{p}(\boldsymbol{X_e}, \boldsymbol{X_t}|\boldsymbol{H_t})}{\mathbf{p}(\boldsymbol{X_e}, \boldsymbol{X_t}|\boldsymbol{H_{nt}})}\right) = \mathbf{log}\left(\frac{\mathbf{p}(\boldsymbol{X_e}, \boldsymbol{X_t})}{\mathbf{p}(\boldsymbol{X_e})\mathbf{p}(\boldsymbol{X_t})}\right)$$

Kenny, “*Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms*”, 2006

Front-end Factor Analysis

- The speaker dependent and independent terms of the JFA model were combined as one total variability term as follows:

$$\boldsymbol{M(s) = M_0 + Ty}$$

Dehak et, al., "*Front-End Factor Analysis for Speaker Verification*", 2010

Front-end Factor Analysis

- The speaker dependent and independent terms of the JFA model were combined as one total variability term as follows:

$$\underbrace{\boldsymbol{M}(\boldsymbol{s})}_{\text{Supervector}} = \boldsymbol{M_0} + \boldsymbol{Ty}$$

Dehak et, al., “*Front-End Factor Analysis for Speaker Verification*”, 2010

Front-end Factor Analysis

- The speaker dependent and independent terms of the JFA model were combined as one total variability term as follows:

$$\underbrace{\boldsymbol{M}}_{\text{Supervector}}(\boldsymbol{s}) = \underbrace{\boldsymbol{M}_{\boldsymbol{0}}}_{\text{Mean component}} + \boldsymbol{T}\boldsymbol{y}$$

Dehak et, al., “*Front-End Factor Analysis for Speaker Verification*”, 2010

Front-end Factor Analysis

- The speaker dependent and independent terms of the JFA model were combined as one total variability term as follows:

$$\boldsymbol{M(s)} = \boldsymbol{M_0} + \boldsymbol{Ty}$$

Supervector

Mean component

Total Variability component

Dehak et, al., “*Front-End Factor Analysis for Speaker Verification*”, 2010

Front-end Factor Analysis

- The speaker dependent and independent terms of the JFA model were combined as one total variability term as follows:

$$\underbrace{\boldsymbol{M}(\boldsymbol{s})}_{\text{Supervector}} = \underbrace{\boldsymbol{M_0}}_{\text{Mean component}} + \underbrace{\boldsymbol{Ty}}_{\text{Total Variability component}} \qquad \text{where } \boldsymbol{y} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I})$$

Front-end Factor Analysis

- The speaker dependent and independent terms of the JFA model were combined as one total variability term as follows:

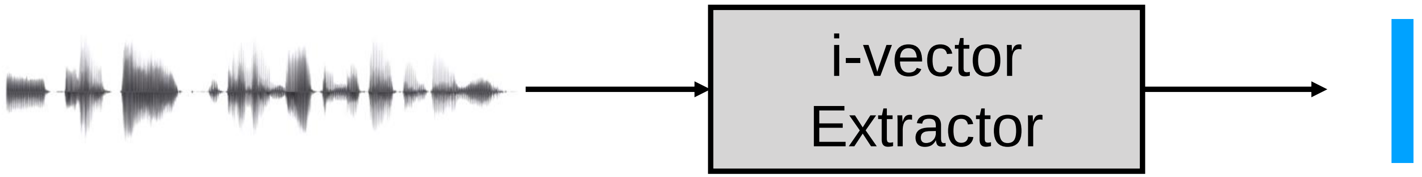
$$M(s) = M_0 + Ty \qquad \text{where } y \sim N(0, I)$$

Supervector

Mean component

Total Variability component

- The MAP estimate of y for a speech segment is called an i-vector, which is a fixed dimensional representation (embedding) of the "total variability" of the segment.



Dehak et, al., "*Front-End Factor Analysis for Speaker Verification*", 2010

Back-End Models

Back-End Models

- Probabilistic Linear Discriminant Analysis (PLDA)

Back-End Models

- Probabilistic Linear Discriminant Analysis (PLDA)

$$\boldsymbol{y} = \underbrace{\boldsymbol{y_s}}_{\text{Speaker Factor}} + \underbrace{\boldsymbol{y_c}}_{\text{Channel Factor}} + \underbrace{\boldsymbol{\epsilon}}_{\text{residual}}$$

Back-End Models

- Probabilistic Linear Discriminant Analysis (PLDA)

$$\boldsymbol{y} = \boldsymbol{y}_s + \boldsymbol{y}_c + \boldsymbol{\epsilon}$$

residual

Speaker Factor

Channel Factor

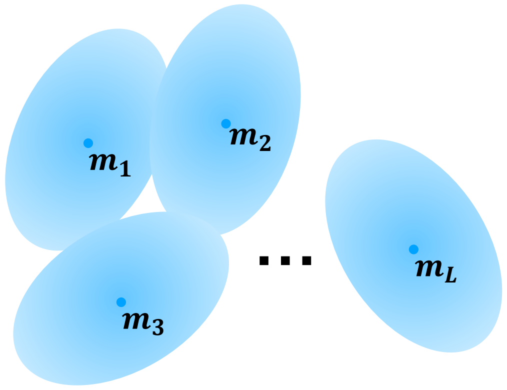
- Gaussian Back-end (GB)

Back-End Models

- Probabilistic Linear Discriminant Analysis (PLDA)

$$\boldsymbol{y} = \underbrace{\boldsymbol{y_s}}_{\text{Speaker Factor}} + \underbrace{\boldsymbol{y_c}}_{\text{Channel Factor}} + \underbrace{\boldsymbol{\epsilon}}_{\text{residual}}$$

- Gaussian Back-end (GB)



Back-End Models

- Probabilistic Linear Discriminant Analysis (PLDA)

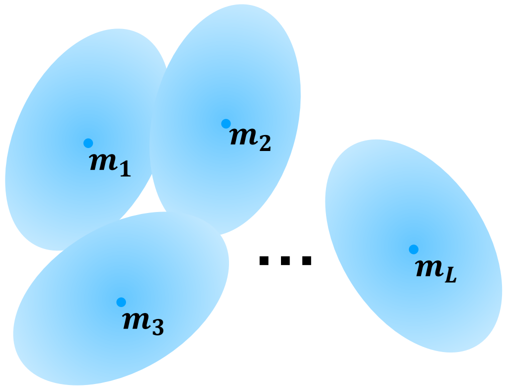
$$\boldsymbol{y} = \boldsymbol{y}_s + \boldsymbol{y}_c + \boldsymbol{\epsilon}$$

residual

Speaker Factor

Channel Factor

- Gaussian Back-end (GB)



- Support Vector Machines

Back-End Models

- Probabilistic Linear Discriminant Analysis (PLDA)

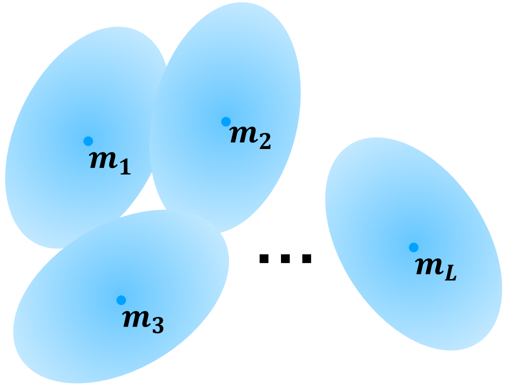
$$\boldsymbol{y} = \boldsymbol{y_s} + \boldsymbol{y_c} + \boldsymbol{\epsilon}$$

residual

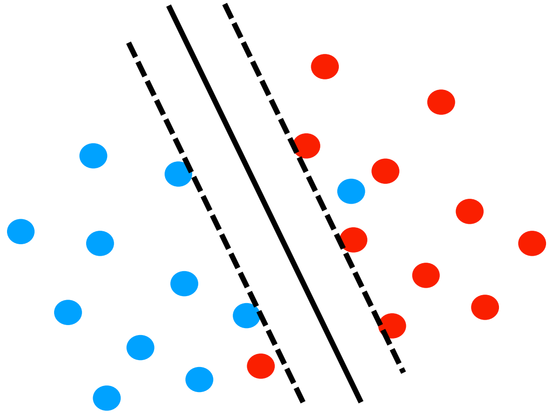
Speaker Factor

Channel Factor

- Gaussian Back-end (GB)



- Support Vector Machines



Back-End Models

- Probabilistic Linear Discriminant Analysis (PLDA)

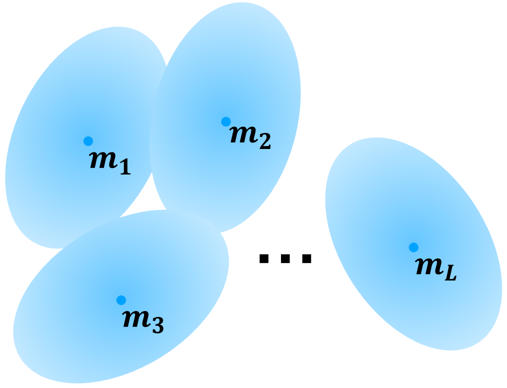
$$\boldsymbol{y} = \boldsymbol{y_s} + \boldsymbol{y_c} + \boldsymbol{\epsilon}$$

residual

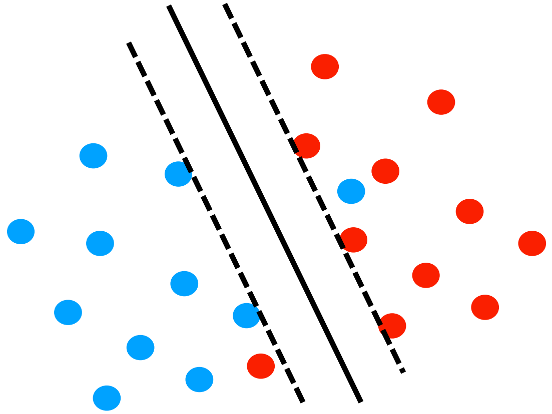
Speaker Factor

Channel Factor

- Gaussian Back-end (GB)



- Support Vector Machines



- Neural Networks

Back-End Models

- Probabilistic Linear Discriminant Analysis (PLDA)

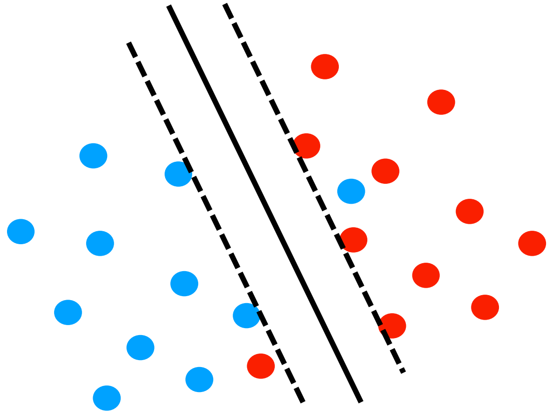
$$\boldsymbol{y} = \boldsymbol{y_s} + \boldsymbol{y_c} + \boldsymbol{\epsilon}$$

residual

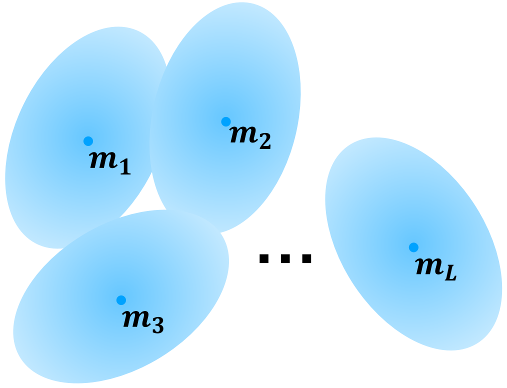
Speaker Factor

Channel Factor

- Support Vector Machines

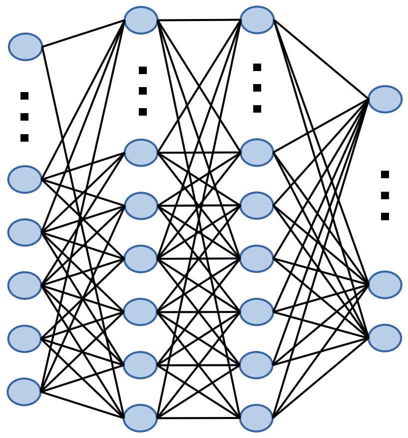


- Gaussian Back-end (GB)



- Neural Networks

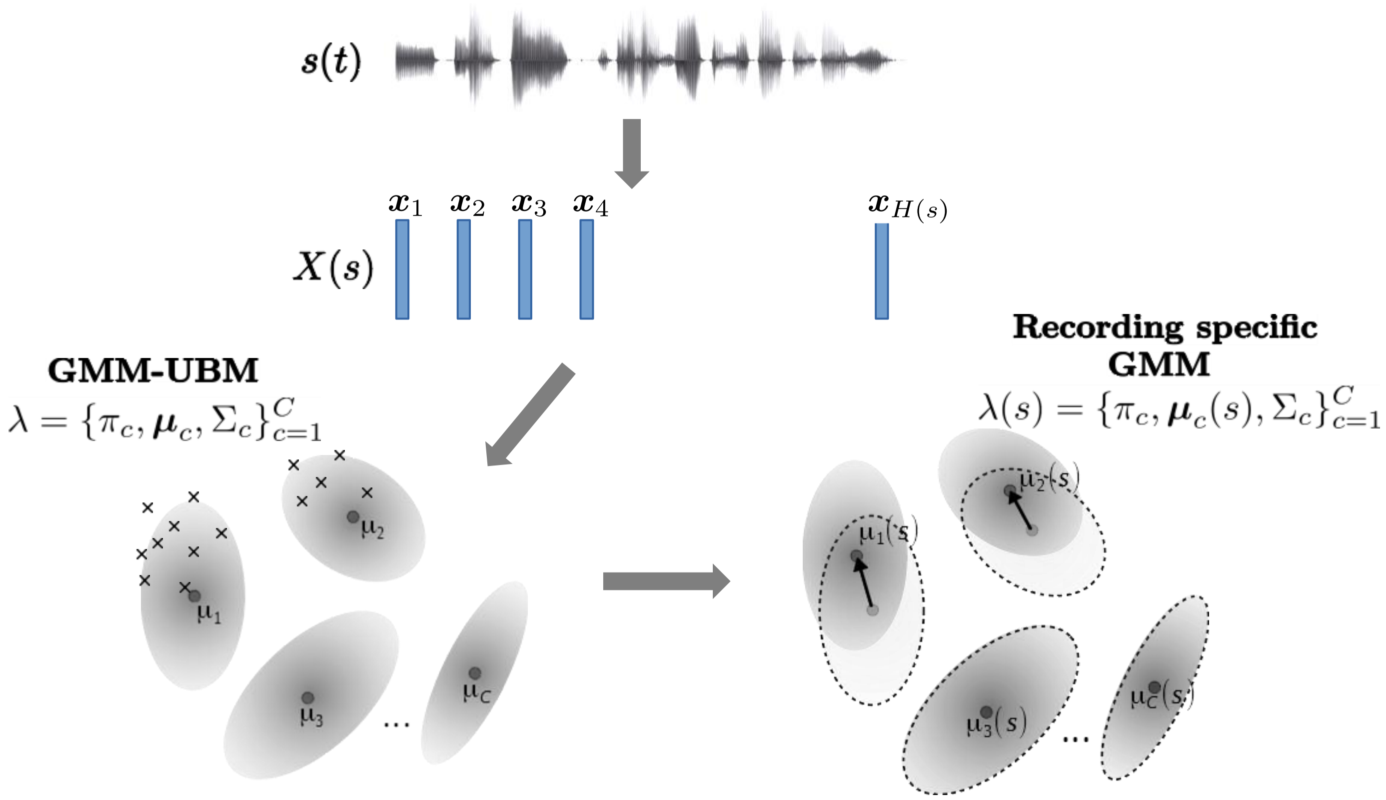
Embeddings



Logits

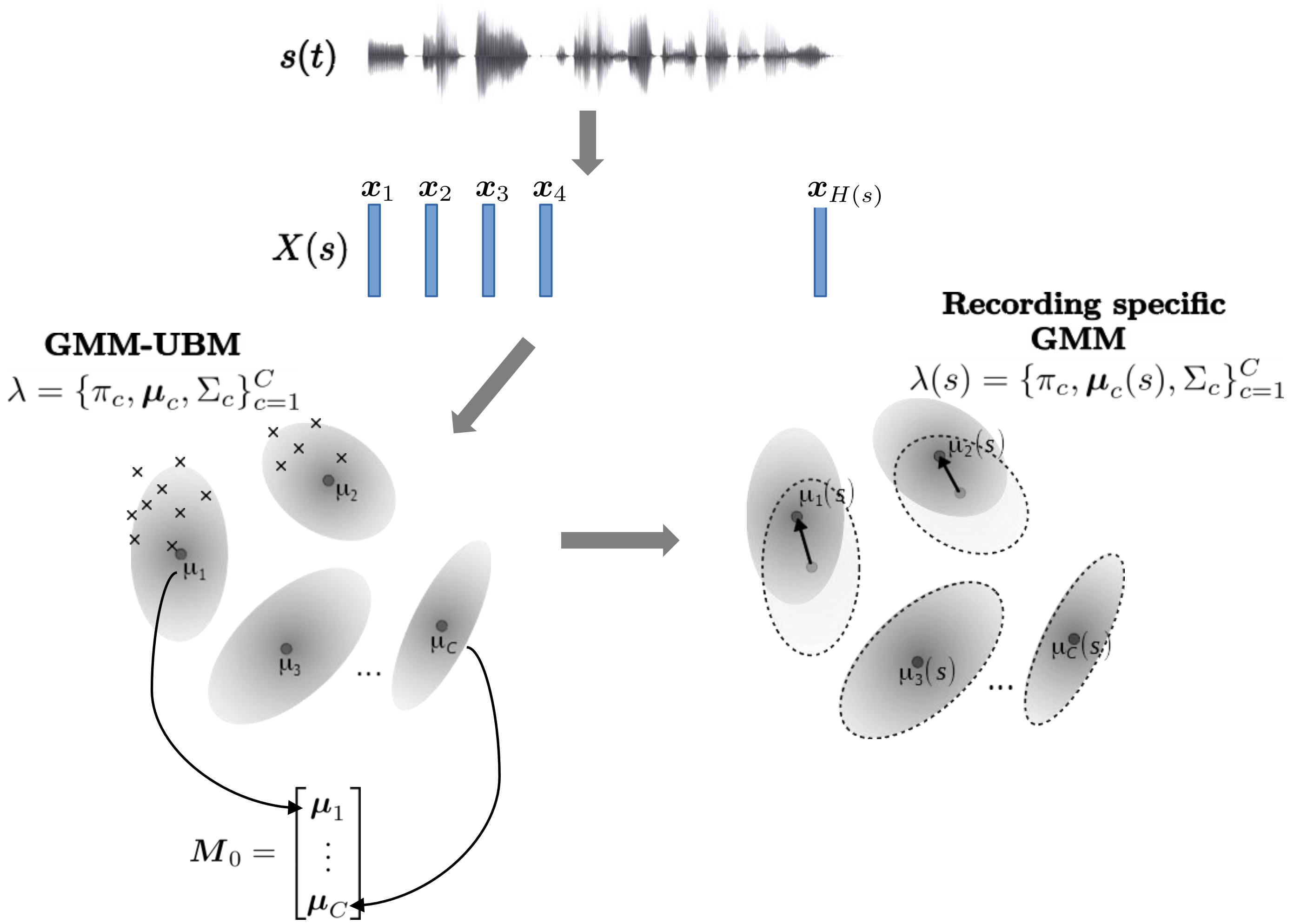
Supervised I-vector Model

Motivation

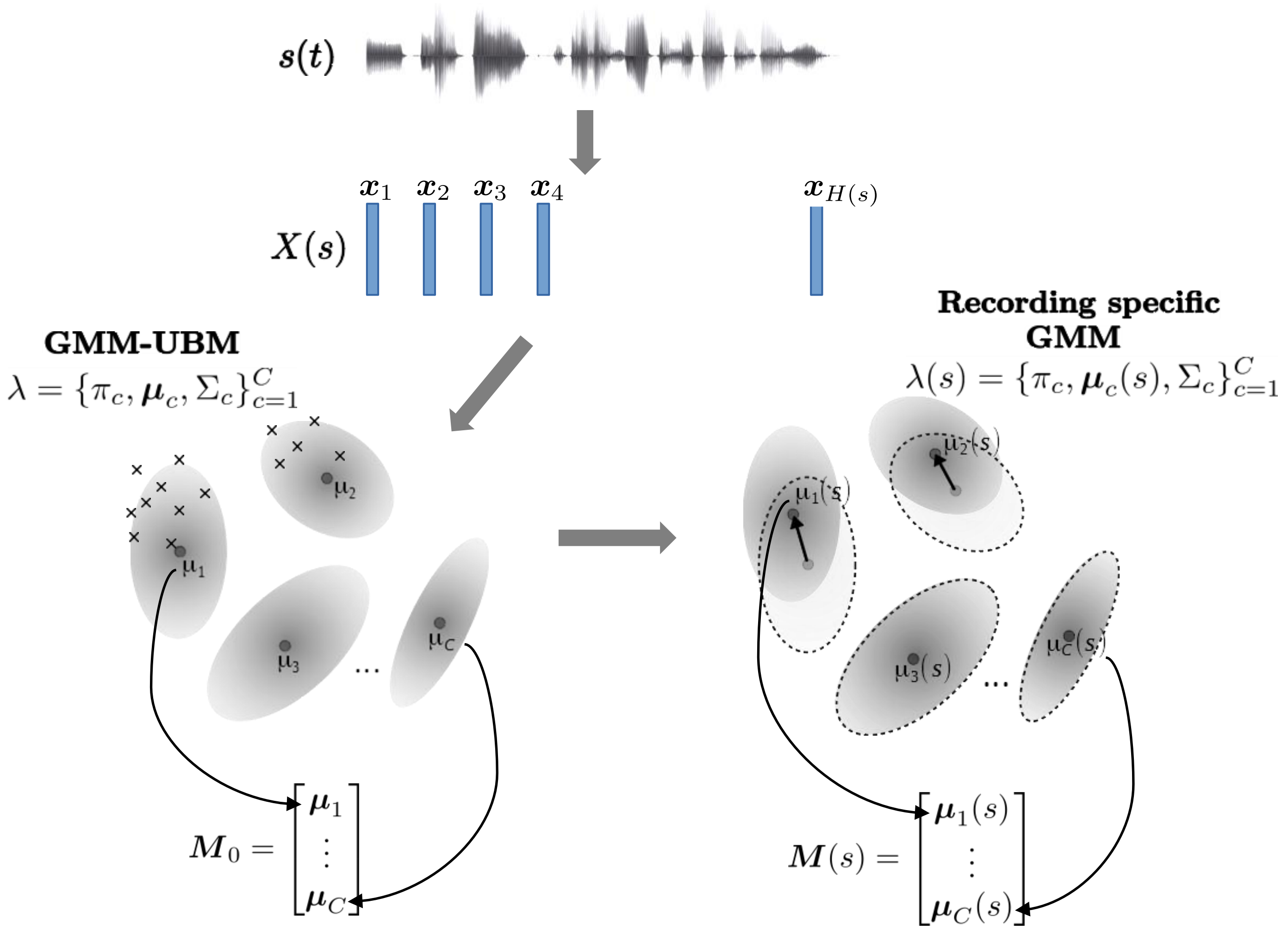


$s(t)$
x_1 x_2 x_3 x_4 $x_{H(s)}$
$X(s)$
GMM-UBM
$\lambda = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^{C}$
Recording specific GMM
$\lambda(s) = \{\pi_c, \boldsymbol{\mu}_c(s), \Sigma_c\}_{c=1}^{C}$
μ_1 μ_2 μ_3 μ_C
$\mu_1(s)$ $\mu_2(s)$ $\mu_3(s)$ $\mu_C(s)$

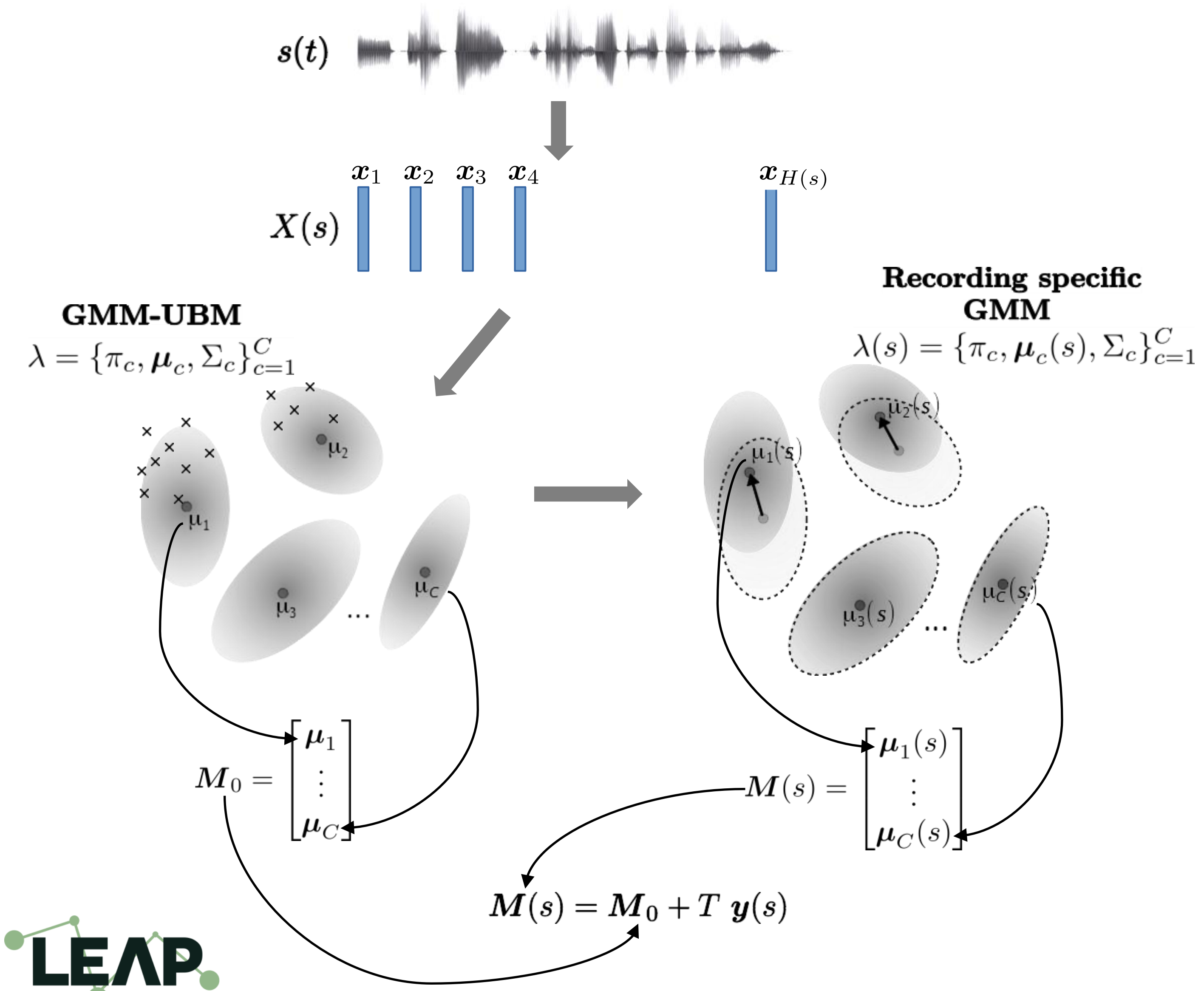
Motivation


$s(t)$
x_1 x_2 x_3 x_4 $x_{H(s)}$
$X(s)$
GMM-UBM
$\lambda = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^{C}$
μ_1 μ_2 μ_3 ... μ_C
$M_0 = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_C \end{bmatrix}$
Recording specific GMM
$\lambda(s) = \{\pi_c, \boldsymbol{\mu}_c(s), \Sigma_c\}_{c=1}^{C}$
$\mu_1(s)$ $\mu_2(s)$ $\mu_3(s)$ ... $\mu_C(s)$

Motivation

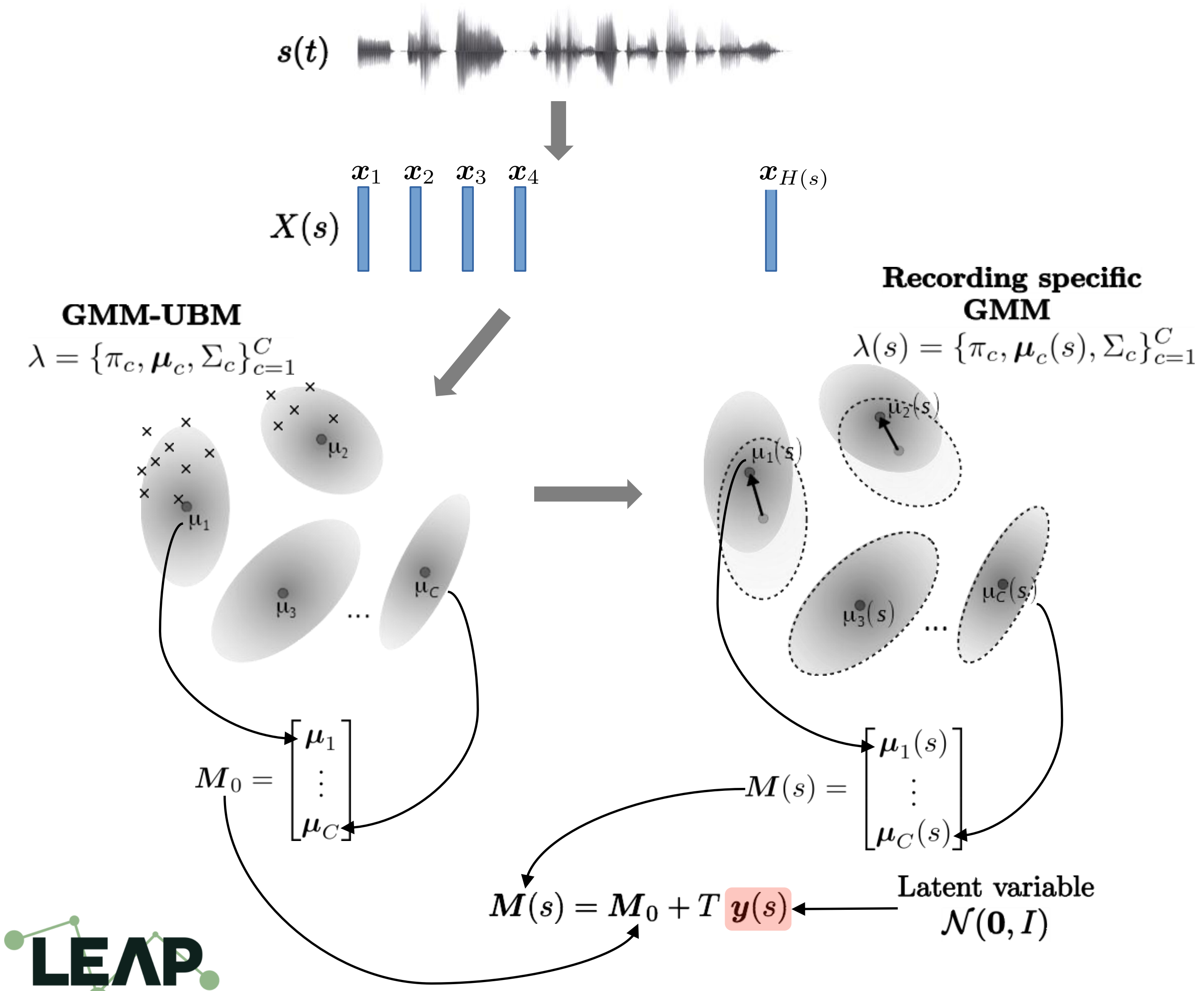

$s(t)$
x_1 x_2 x_3 x_4 $x_{H(s)}$
$X(s)$
GMM-UBM
$\lambda = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^{C}$
Recording specific GMM
$\lambda(s) = \{\pi_c, \boldsymbol{\mu}_c(s), \Sigma_c\}_{c=1}^{C}$
$M_0 = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_C \end{bmatrix}$
$M(s) = \begin{bmatrix} \boldsymbol{\mu}_1(s) \\ \vdots \\ \boldsymbol{\mu}_C(s) \end{bmatrix}$

Motivation



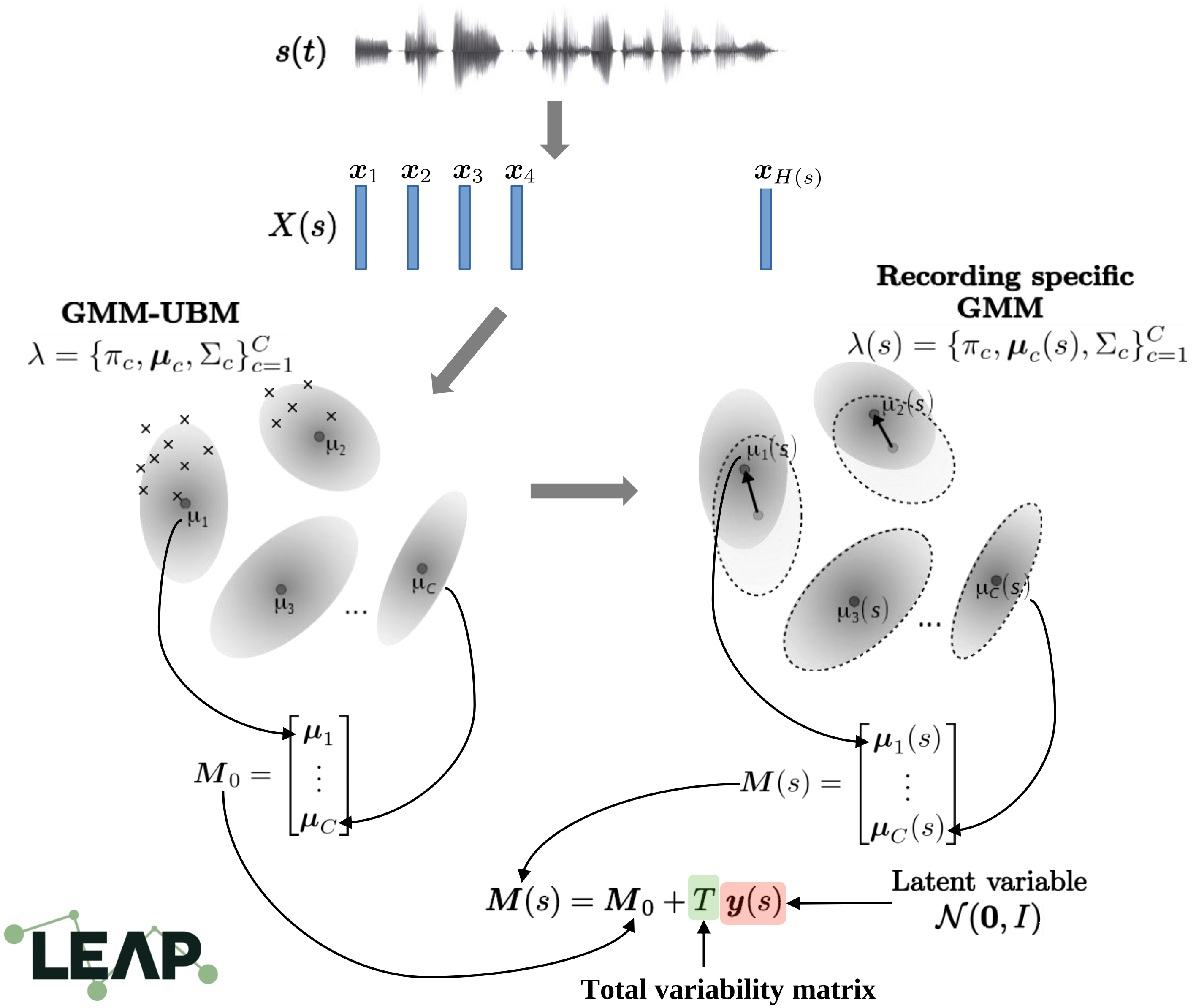
$s(t)$
x_1 x_2 x_3 x_4 $x_{H(s)}$
$X(s)$
GMM-UBM
$\lambda = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^{C}$
μ_1 μ_2 μ_3 μ_C
Recording specific GMM
$\lambda(s) = \{\pi_c, \boldsymbol{\mu}_c(s), \Sigma_c\}_{c=1}^{C}$
$\mu_1(s)$ $\mu_2(s)$ $\mu_3(s)$ $\mu_C(s)$
$\boldsymbol{M}_0 = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_C \end{bmatrix}$
$\boldsymbol{M}(s) = \begin{bmatrix} \boldsymbol{\mu}_1(s) \\ \vdots \\ \boldsymbol{\mu}_C(s) \end{bmatrix}$
$\boldsymbol{M}(s) = \boldsymbol{M}_0 + T\,\boldsymbol{y}(s)$

Motivation



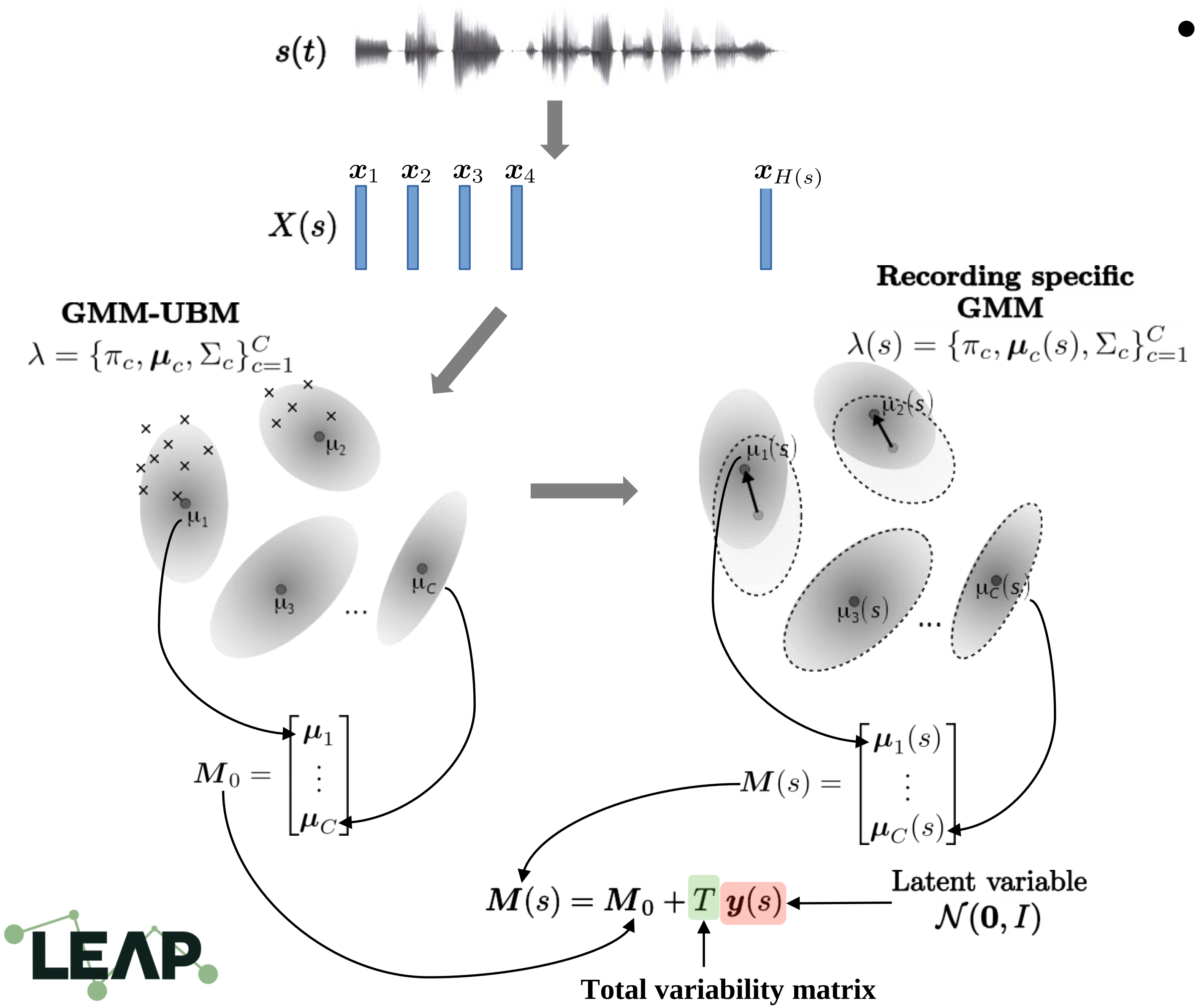
$s(t)$
x_1 x_2 x_3 x_4 $x_{H(s)}$
$X(s)$
GMM-UBM
$\lambda = \{\pi_c, \mu_c, \Sigma_c\}_{c=1}^{C}$
Recording specific GMM
$\lambda(s) = \{\pi_c, \mu_c(s), \Sigma_c\}_{c=1}^{C}$
$M_0 = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_C \end{bmatrix}$
$M(s) = \begin{bmatrix} \mu_1(s) \\ \vdots \\ \mu_C(s) \end{bmatrix}$
$M(s) = M_0 + T\, y(s)$
Latent variable
$\mathcal{N}(\mathbf{0}, I)$

Motivation

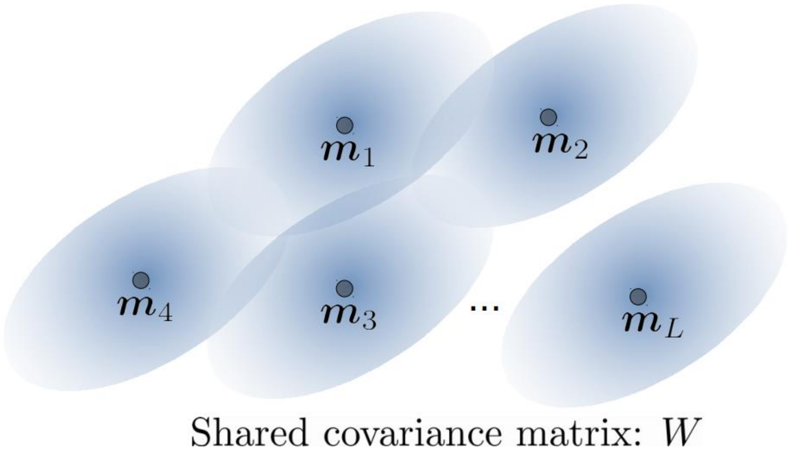


$s(t)$
x_1 x_2 x_3 x_4 $x_{H(s)}$
$X(s)$
GMM-UBM
$\lambda = \{\pi_c, \mu_c, \Sigma_c\}_{c=1}^{C}$
μ_1 μ_2 μ_3 μ_C ...
Recording specific GMM
$\lambda(s) = \{\pi_c, \mu_c(s), \Sigma_c\}_{c=1}^{C}$
$\mu_1(s)$ $\mu_2(s)$ $\mu_3(s)$ $\mu_C(s)$ ...
$M_0 = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_C \end{bmatrix}$
$M(s) = \begin{bmatrix} \mu_1(s) \\ \vdots \\ \mu_C(s) \end{bmatrix}$
$M(s) = M_0 + T\, y(s)$
Latent variable
$\mathcal{N}(0, I)$
Total variability matrix

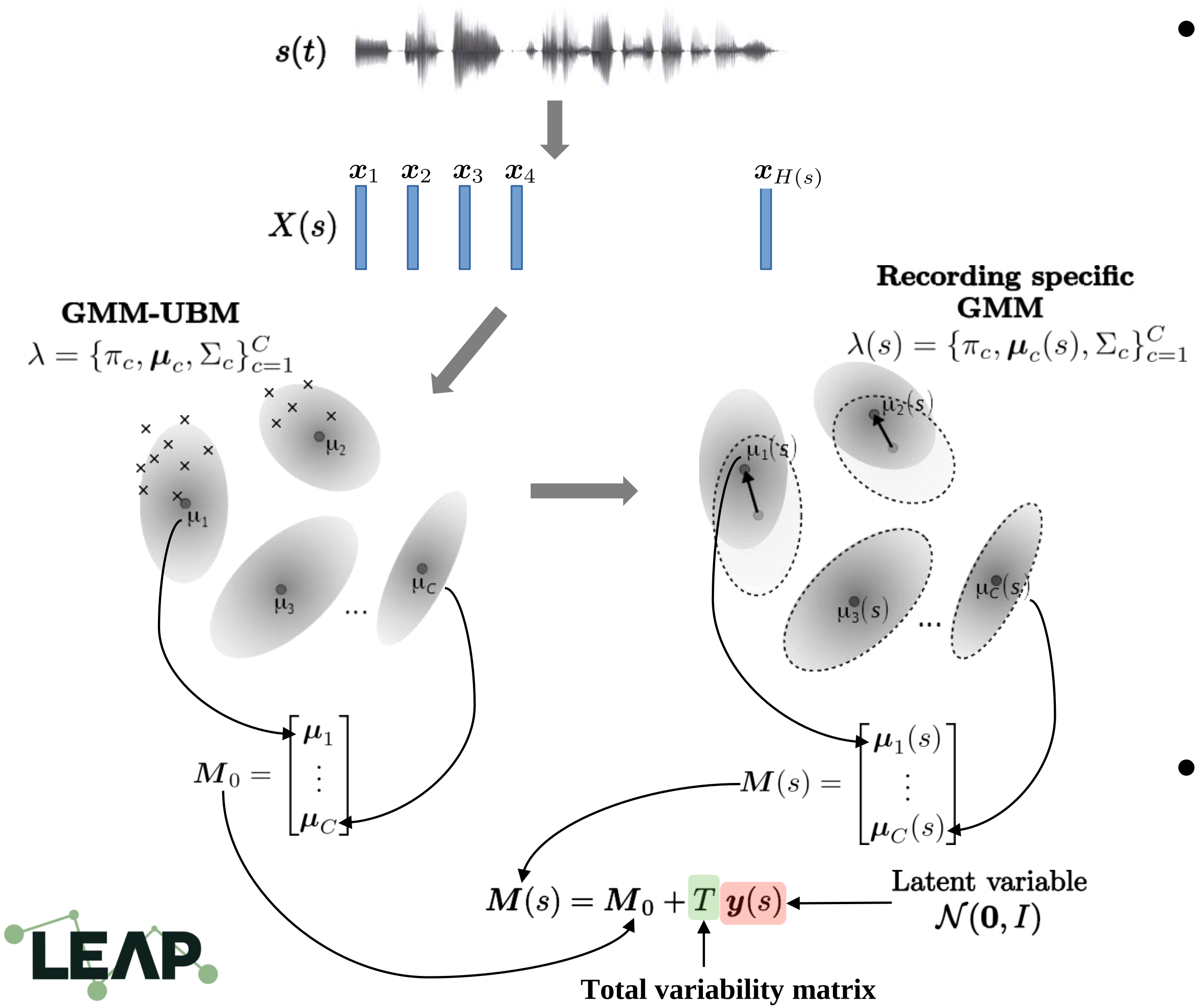
Motivation



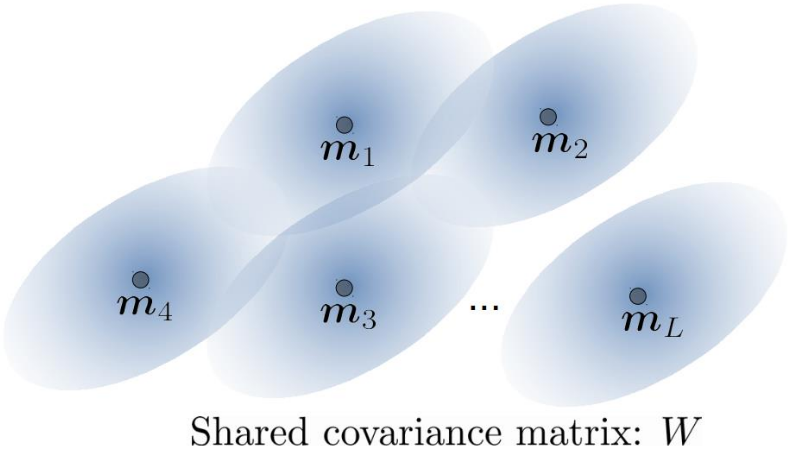
- The state-of-the-art language recognition system (till 2017) was the i-vector Gaussian Back-end approach.



Motivation



- The state-of-the-art language recognition system (till 2017) was the i-vector Gaussian Back-end approach.



- The label log-likelihoods are computed as

$$\log \mathrm{p}(\boldsymbol{y} \mid l) = \boldsymbol{y}^{\intercal} W^{-1} \boldsymbol{m}_l - \frac{1}{2} \boldsymbol{m}_l^{\intercal} W^{-1} \boldsymbol{m}_l$$

Motivation

- i-vectors are unsupervised embeddings that capture the total variability of an utterance from the GMM-UBM.

Motivation

- i-vectors are unsupervised embeddings that capture the total variability of an utterance from the GMM-UBM.

- For training a language recognition system, the labels are utilized only in the back-end modeling stage. Lot of relevant information may be lost in i-vector extraction.

Motivation

- i-vectors are unsupervised embeddings that capture the total variability of an utterance from the GMM-UBM.

- For training a language recognition system, the labels are utilized only in the back-end modeling stage. Lot of relevant information may be lost in i-vector extraction.

- Can we incorporate the label information into the model, making it supervised?

Motivation

- i-vectors are unsupervised embeddings that capture the total variability of an utterance from the GMM-UBM.
- For training a language recognition system, the labels are utilized only in the back-end modeling stage. Lot of relevant information may be lost in i-vector extraction.
- Can we incorporate the label information into the model, making it supervised?
- i-vectors have a standard normal assumption in the front-end, but a GMM assumption in the back-end.

Motivation

- i-vectors are unsupervised embeddings that capture the total variability of an utterance from the GMM-UBM.
- For training a language recognition system, the labels are utilized only in the back-end modeling stage. Lot of relevant information may be lost in i-vector extraction.
- Can we incorporate the label information into the model, making it supervised?
- i-vectors have a standard normal assumption in the front-end, but a GMM assumption in the back-end.
- Can we train the front-end i-vector model with a mixture Gaussian prior, making the assumptions consistent with the back-end?

The s-vector model

- The proposed supervised i-vector (s-vector) model is expressed as

$$\boldsymbol{M}(s) = \boldsymbol{M}_0 + T\,\boldsymbol{y}(s)$$

The s-vector model

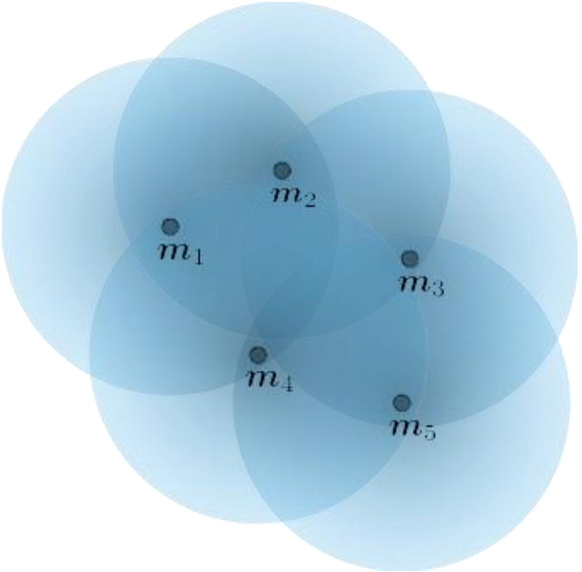
- The proposed supervised i-vector (s-vector) model is expressed as

$$\boldsymbol{M}(s) = \boldsymbol{M}_0 + T\,\boldsymbol{y}(s)$$

- Where, the prior of latent variable $\boldsymbol{y}(s)$ for class l is given by:

$$p(\boldsymbol{y}(s) | l(s) = l) = \mathcal{N}(\boldsymbol{y}(s); \boldsymbol{m}_l, I)$$

$$p\big(\boldsymbol{y}(s)\big) = \sum_{l=1}^{L} p(l)\mathcal{N}(\boldsymbol{y}(s); \boldsymbol{m}_l, I)$$



The s-vector model

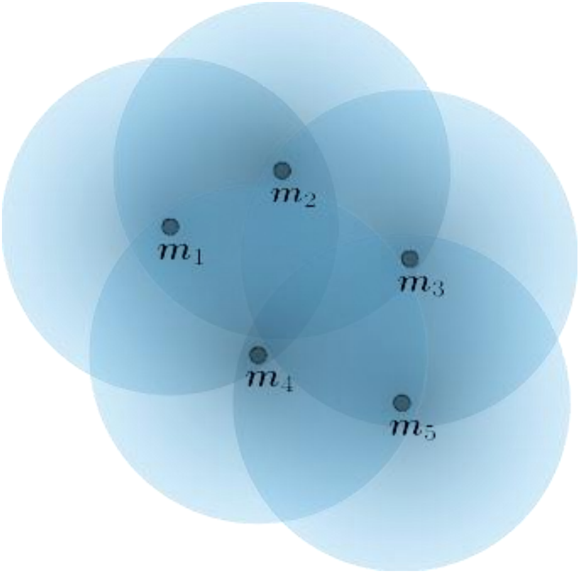
- The proposed supervised i-vector (s-vector) model is expressed as

$$\boldsymbol{M}(s) = \boldsymbol{M}_0 + T\,\boldsymbol{y}(s)$$

- Where, the prior of latent variable $\boldsymbol{y}(s)$ for class l is given by:

$$p(\boldsymbol{y}(s)|l(s) = l) = \mathcal{N}(\boldsymbol{y}(s); \boldsymbol{m}_l, I)$$

$$p\big(\boldsymbol{y}(s)\big) = \sum_{l=1}^{L} p(l)\mathcal{N}(\boldsymbol{y}(s); \boldsymbol{m}_l, I)$$



- The parameters of the s-vector model are denoted as:

$$\Theta = \{T, \boldsymbol{m}_1, \dots, \boldsymbol{m}_L\}$$

Training the *s*-vector model

- For the (unsupervised) i-vector model, the estimation problem is formulated as:

$$T = \underset{T}{\operatorname{argmax}} \sum_{s} \log p_T(X(s))$$

Training the s-vector model

- For the (unsupervised) i-vector model, the estimation problem is formulated as:

$$T = \underset{T}{\operatorname{argmax}} \sum_{s} \log p_T(X(s))$$

$$T = \underset{T}{\operatorname{argmax}} \sum_{s} \log \int_{\boldsymbol{y}} p_T(X(s), \boldsymbol{y}) \, d\boldsymbol{y}$$

Training the s-vector model

- For the (unsupervised) i-vector model, the estimation problem is formulated as:

$$T = \underset{T}{\operatorname{argmax}} \sum_{s} \log p_T(X(s))$$

$$T = \underset{T}{\operatorname{argmax}} \sum_{s} \log \int_{\boldsymbol{y}} p_T(X(s), \boldsymbol{y})\, d\boldsymbol{y}$$

- For our s-vector model, we reformulate the estimation problem as

$$\Theta = \underset{\Theta}{\operatorname{argmax}} \sum_{s} \log p_\Theta(X(s), l(s))$$

$$\Theta = \underset{\Theta}{\operatorname{argmax}} \sum_{s} \log \int_{\boldsymbol{y}} p_\Theta(X(s), l(s), \boldsymbol{y})\, d\boldsymbol{y}$$

Training the s-vector model

- For the (unsupervised) i-vector model, the estimation problem is formulated as:

$$T = \underset{T}{\operatorname{argmax}} \sum_{s} \log p_T(X(s))$$

$$T = \underset{T}{\operatorname{argmax}} \sum_{s} \log \int_{\boldsymbol{y}} p_T(X(s), \boldsymbol{y})\, d\boldsymbol{y}$$

- For our s-vector model, we reformulate the estimation problem as

$$\Theta = \underset{\Theta}{\operatorname{argmax}} \sum_{s} \log \boxed{p_\Theta(X(s), l(s))}$$

Joint likelihood of features & Labels

$$\Theta = \underset{\Theta}{\operatorname{argmax}} \sum_{s} \log \int_{\boldsymbol{y}} p_\Theta(X(s), l(s), \boldsymbol{y})\, d\boldsymbol{y}$$

Training the s-vector model

Training the s-vector model

- We use the Expectation-Maximization (EM) algorithm to solve for the model parameters.

Training the *s*-vector model

- We use the Expectation-Maximization (EM) algorithm to solve for the model parameters.

- Expectation (E) Step:

Training the s-vector model

- We use the Expectation-Maximization (EM) algorithm to solve for the model parameters.

- Expectation (E) Step:

$$Q\left(T \middle| T^{(t)}\right) = \sum_{s} \mathbb{E}_{y(s)|X(s),T^{(t)}} \log p_T(X(s), y(s))$$

Training the s-vector model

- We use the Expectation-Maximization (EM) algorithm to solve for the model parameters.
- Expectation (E) Step:

$$Q\left(T \middle| T^{(t)}\right) = \sum_{s} \mathbb{E}_{y(s)|X(s),T^{(t)}} \log p_T(X(s), y(s))$$

- Maximization (M) Step:

Training the s-vector model

- We use the Expectation-Maximization (EM) algorithm to solve for the model parameters.

- Expectation (E) Step:

$$Q\left(T \middle| T^{(t)}\right) = \sum_{s} \mathbb{E}_{y(s)|X(s),T^{(t)}} \log p_T(X(s), y(s))$$

- Maximization (M) Step:

$$T^{(t+1)} = \underset{T}{\operatorname{argmax}}\, Q\left(T \middle| T^{(t)}\right)$$

LEAP

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Training the s-vector model

- We use the Expectation-Maximization (EM) algorithm to solve for the model parameters.

- Expectation (E) Step:

$$Q\left(T \middle| T^{(t)}\right) = \sum_{s} \mathbb{E}_{y(s)|X(s),T^{(t)}} \log p_T(X(s), y(s))$$

$$Q\left(\Theta \middle| \Theta^{(t)}\right) = \sum_{s} \mathbb{E}_{y(s)|X(s),l(s),\Theta^{(t)}} \log p_\Theta(X(s), l(s), y(s))$$

- Maximization (M) Step:

$$T^{(t+1)} = \underset{T}{\operatorname{argmax}}\, Q\left(T \middle| T^{(t)}\right)$$

LEAP

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Training the s-vector model

- We use the Expectation-Maximization (EM) algorithm to solve for the model parameters.

- Expectation (E) Step:

$$Q\left(T \middle| T^{(t)}\right) = \sum_{s} \mathbb{E}_{y(s)|X(s),T^{(t)}} \log p_T(X(s), y(s))$$

$$Q\left(\Theta \middle| \Theta^{(t)}\right) = \sum_{s} \mathbb{E}_{y(s)|X(s),l(s),\Theta^{(t)}} \log p_\Theta(X(s), l(s), y(s))$$

- Maximization (M) Step:

$$T^{(t+1)} = \underset{T}{\operatorname{argmax}}\, Q\left(T \middle| T^{(t)}\right)$$

$$\Theta^{(t+1)} = \underset{\Theta}{\operatorname{argmax}}\, Q\left(\Theta \middle| \Theta^{(t)}\right)$$

Training the s-vector model

- E – Step: We compute the following quantities:

$$\mathrm{N}_c(s) = \sum_i p(c|\boldsymbol{x}_i(s))$$

$$\mathrm{N}(s) = \begin{pmatrix} \mathrm{N}_1(s) & & 0 \\ & \ddots & \\ 0 & & \mathrm{N}_C(s) \end{pmatrix}$$

$$\mathbf{F}_c(s) = \sum_i \boldsymbol{x}_i(s)\mathrm{p}(c|x_i(s))$$

$$\mathbf{F}(s) = \begin{pmatrix} \mathbf{F}_1(s) \\ \vdots \\ \mathbf{F}_c(s) \end{pmatrix}$$

Training the s-vector model

- E – Step: We compute the following quantities:

$$\mathrm{N}_c(s) = \sum_i p(c|\boldsymbol{x}_i(s))$$

$$\mathrm{N}(s) = \begin{pmatrix} \mathrm{N}_1(s) & & 0 \\ & \ddots & \\ 0 & & \mathrm{N}_C(s) \end{pmatrix}$$

$$\mathbf{F}_c(s) = \sum_i \boldsymbol{x}_i(s)\mathrm{p}(c|x_i(s))$$

$$\mathbf{F}(s) = \begin{pmatrix} \mathbf{F}_1(s) \\ \vdots \\ \mathbf{F}_c(s) \end{pmatrix}$$

$$\mathcal{L}^{(t)}(s) = I + {T^{(t)}}^{\mathrm{T}} \Sigma^{-1} \mathrm{N}(s) T^{(t)}$$

Training the s-vector model

- E – Step: We compute the following quantities:

$$\mathrm{N}_c(s) = \sum_i p(c|\boldsymbol{x}_i(s))$$

$$\mathrm{N}(s) = \begin{pmatrix} \mathrm{N}_1(s) & & 0 \\ & \ddots & \\ 0 & & \mathrm{N}_C(s) \end{pmatrix}$$

$$\mathbf{F}_c(s) = \sum_i \boldsymbol{x}_i(s)\mathrm{p}(c|x_i(s))$$

$$\mathbf{F}(s) = \begin{pmatrix} \mathbf{F}_1(s) \\ \vdots \\ \mathbf{F}_c(s) \end{pmatrix}$$

$$\mathcal{L}^{(t)}(s) = I + {T^{(t)}}^{\mathrm{T}} \Sigma^{-1} \mathrm{N}(s) T^{(t)}$$

$$\hat{\boldsymbol{y}}_{l(s)}^{(t)}(s) = \mathcal{L}^{(t)}(s)^{-1}\left(\boldsymbol{m}_{l(s)}^{(t)} + {T^{(t)}}^{T} \Sigma^{-1} \mathbf{F}(s)\right)$$

$$E_{\boldsymbol{yy},l(s)}^{(t)}(s) = \mathcal{L}^{(t)}(s)^{-1} + \hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)\, \hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)^{\mathrm{T}}$$

Training the s-vector model

- E – Step: We compute the following quantities:

$$\mathrm{N}_c(s) = \sum_i p(c|\boldsymbol{x}_i(s))$$

$$\mathrm{N(s)} = \begin{pmatrix} \mathrm{N}_1(s) & & 0 \\ & \ddots & \\ 0 & & \mathrm{N}_C(s) \end{pmatrix}$$

$$\mathbf{F}_c(s) = \sum_i \boldsymbol{x}_i(s)\mathrm{p}(c|x_i(s))$$

$$\mathbf{F}(\mathrm{s}) = \begin{pmatrix} \mathbf{F}_1(s) \\ \vdots \\ \mathbf{F}_c(s) \end{pmatrix}$$

$$\mathcal{L}^{(t)}(s) = I + {T^{(t)}}^{\mathrm{T}} \Sigma^{-1} \mathrm{N(s)} T^{(t)}$$

$$\widehat{\boldsymbol{y}}_{l(s)}^{(t)}(s) = \mathcal{L}^{(t)}(s)^{-1}\left(\boldsymbol{m}_{l(s)}^{(t)} + {T^{(t)}}^{T} \Sigma^{-1} \mathbf{F}(\mathrm{s})\right)$$

$$E_{\boldsymbol{yy},l(s)}^{(t)}(s) = \mathcal{L}^{(t)}(s)^{-1} + \widehat{\boldsymbol{y}}_{l(s)}^{(t)}(s)\, \widehat{\boldsymbol{y}}_{l(s)}^{(t)}(s)^{\mathrm{T}}$$

$$Q\left(\Theta|\Theta^{(t)}\right) = \sum_s \left[\mathrm{tr}\{T^{\mathrm{T}}\Sigma^{-1}\boldsymbol{F}(s)\widehat{\boldsymbol{y}}_{l(s)}^{t}(s)^{\mathrm{T}}\} - \tfrac{1}{2}\mathrm{tr}\left\{\Sigma^{-1}N(s)TE_{yy,l(s)}^{(t)}(s)T^{\mathrm{T}}\right\}\right] + \sum_l \sum_{\substack{s \\ l(s)=l}} \left(\widehat{\boldsymbol{y}}_{l(s)}^{(t)}(s)^{\mathrm{T}}\boldsymbol{m}_l - \tfrac{1}{2}\boldsymbol{m}_l\boldsymbol{m}_l^{\mathrm{T}}\right)$$

Training the s-vector model

- M – Step: Maximize the convex Q function to solve for the parameters.

Training the s-vector model

- M – Step: Maximize the convex Q function to solve for the parameters.

- Update equations for T :

$$\boxed{T_c^{(t+1)} = \left(\sum_s N_c(s) E^{(t)}_{\boldsymbol{yy},l(s)}(s)\right)^{-1} \sum_s F_c(s)\, \hat{\boldsymbol{y}}^{(t)}_{l(s)}\,(s)^{\mathrm{T}}}$$

Training the s-vector model

- M – Step: Maximize the convex Q function to solve for the parameters.

- Update equations for T :

$$\boxed{T_c^{(t+1)} = \left(\sum_s N_c(s) E^{(t)}_{\boldsymbol{yy},l(s)}(s)\right)^{-1} \sum_s F_c(s)\, \hat{\boldsymbol{y}}^{(t)}_{l(s)}(s)^{\mathrm{T}}}$$

- Update equations for $\boldsymbol{m}_l$:

$$\boxed{\boldsymbol{m}_l^{(t+1)} = \sum_{\substack{s \\ l(s)=l}} \hat{\boldsymbol{y}}^{(t)}_{l(s)}(s)}$$

Extracting i-vectors / s-vectors

Extracting i-vectors / s-vectors

Extracting i-vectors / s-vectors

- In the i-vector model, the **posterior distribution** of the latent variable $\boldsymbol{y}$, for a recording s is given by:

Extracting i-vectors / s-vectors

- In the i-vector model, the **posterior distribution** of the latent variable $\boldsymbol{y}$, for a recording s is given by:

$$p(\boldsymbol{y}(s)|X(s)) = \mathcal{N}(\mathcal{L}(s)^{-1}T^T\Sigma^{-1}\mathbf{F}(\mathrm{s})\,,\ \mathcal{L}(s)^{-1})$$

Extracting i-vectors / s-vectors

- In the i-vector model, the **posterior distribution** of the latent variable $\boldsymbol{y}$, for a recording s is given by:

$$p(\boldsymbol{y}(s)|X(s)) = \mathcal{N}(\mathcal{L}(s)^{-1}T^{T}\Sigma^{-1}\mathbf{F}(s)\,,\ \mathcal{L}(s)^{-1})$$

Extracting i-vectors / s-vectors

- In the i-vector model, the **posterior distribution** of the latent variable $\boldsymbol{y}$, for a recording s is given by:

$$p(\boldsymbol{y}(s)|X(s)) = \mathcal{N}(\mathcal{L}(s)^{-1}T^T\Sigma^{-1}\mathbf{F}(\mathrm{s})\,,\ \mathcal{L}(s)^{-1})$$

Extracting i-vectors / s-vectors

- In the i-vector model, the **posterior distribution** of the latent variable $\boldsymbol{y}$, for a recording s is given by:

$$p(\boldsymbol{y}(s)|X(s)) = \mathcal{N}(\underbrace{\mathcal{L}(s)^{-1}T^{T}\Sigma^{-1}\mathbf{F}(s)}_{\hat{\boldsymbol{y}}(s)}, \mathcal{L}(s)^{-1})$$

$\hat{\boldsymbol{y}}(s) \leftarrow$ **i-vector** **(MAP estimate)**

Extracting i-vectors / s-vectors

- In the i-vector model, the **posterior distribution** of the latent variable $\boldsymbol{y}$, for a recording s is given by:

$$p(\boldsymbol{y}(s)|X(s)) = \mathcal{N}(\underbrace{\mathcal{L}(s)^{-1}T^T\Sigma^{-1}\mathbf{F}(s)}_{\hat{\boldsymbol{y}}(s)}\,,\ \mathcal{L}(s)^{-1})$$

$\hat{\boldsymbol{y}}(s) \longleftarrow$ **i-vector** **(MAP estimate)**

- In the s-vector model, the **posterior distribution** of $\boldsymbol{y}$ is given by

$$p(\boldsymbol{y}(s)|X(s)) = \sum_{l} p(l|X(s))\mathcal{N}(\mathcal{L}(s)^{-1}(m_l + T^T\Sigma^{-1}\mathbf{F}(s))\,,\ \mathcal{L}(s)^{-1})$$

LEAP

INDIAN INSTITUTE OF SCIENCE भारतीय विज्ञान संस्थान

Extracting i-vectors / s-vectors

- In the i-vector model, the **posterior distribution** of the latent variable $\boldsymbol{y}$, for a recording s is given by:

$$p(\boldsymbol{y}(s)|X(s)) = \mathcal{N}(\underbrace{\mathcal{L}(s)^{-1}T^T\Sigma^{-1}\mathbf{F}(s)}_{\hat{\boldsymbol{y}}(s)}, \mathcal{L}(s)^{-1})$$

$\hat{\boldsymbol{y}}(s) \leftarrow$ **i-vector** **(MAP estimate)**

- In the s-vector model, the **posterior distribution** of $\boldsymbol{y}$ is given by

$$p(\boldsymbol{y}(s)|X(s)) = \sum_l p(l|X(s))\mathcal{N}(\underbrace{\mathcal{L}(s)^{-1}(m_l + T^T\Sigma^{-1}\mathbf{F}(s))}_{\hat{\boldsymbol{y}}_l(s)}, \mathcal{L}(s)^{-1})$$

$\hat{\boldsymbol{y}}_l(s) \leftarrow$ **Class conditioned s-vector**

Extracting i-vectors / s-vectors

- Approach 1: PCA for dimensionality reduction (PCA s-vectors)

$$\begin{pmatrix} \hat{y}_1(s) \\ \hat{y}_2(s) \\ \vdots \\ \hat{y}_L(s) \end{pmatrix} \longrightarrow \boxed{\begin{array}{c}\text{PCA}\\ \text{Transformation}\end{array}} \longrightarrow \hat{\boldsymbol{y}}_{\boldsymbol{PCA}}(s)$$

Extracting i-vectors / s-vectors

- Approach 1: PCA for dimensionality reduction (PCA s-vectors)

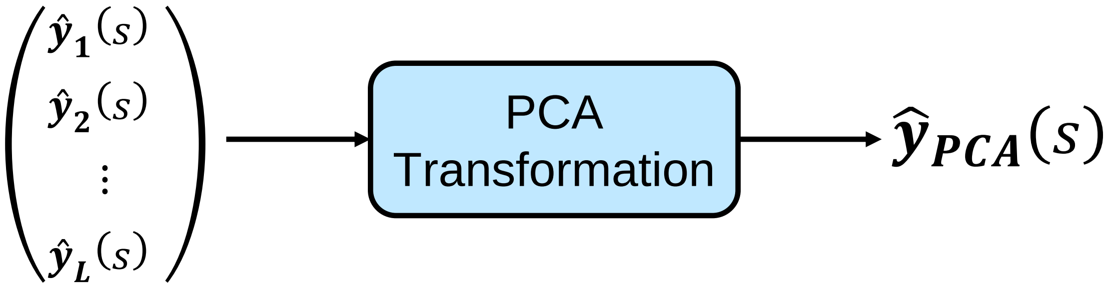
$$\begin{pmatrix} \hat{y}_1(s) \\ \hat{y}_2(s) \\ \vdots \\ \hat{y}_L(s) \end{pmatrix} \longrightarrow \boxed{\begin{array}{c}\text{PCA}\\ \text{Transformation}\end{array}} \longrightarrow \hat{\boldsymbol{y}}_{\boldsymbol{PCA}}(s)$$

- Approach 2: Average class conditioned s-vectors (Avg s-vectors)

$$\hat{\boldsymbol{y}}_{\boldsymbol{avg}}(s) = \frac{1}{L}\sum_{l} \hat{\boldsymbol{y}}_l(s)$$

Extracting i-vectors / s-vectors

- Approach 1: PCA for dimensionality reduction (PCA s-vectors)



- Approach 2: Average class conditioned s-vectors (Avg s-vectors)

$$\hat{\boldsymbol{y}}_{\boldsymbol{avg}}(s) = \frac{1}{L}\sum_{l} \hat{\boldsymbol{y}}_l(s)$$

- Approach 3: Minimum mean squared error s-vectors (MMSE s-vectors)

$$\hat{\boldsymbol{y}}_{\boldsymbol{MMSE}}(s) = \sum_{l} p(l|X(s))\hat{\boldsymbol{y}}_l(s)$$

Extracting i-vectors / s-vectors

- Approach 1: PCA for dimensionality reduction (PCA s-vectors)

$$\begin{pmatrix} \hat{y}_1(s) \\ \hat{y}_2(s) \\ \vdots \\ \hat{y}_L(s) \end{pmatrix} \longrightarrow \boxed{\text{PCA Transformation}} \longrightarrow \hat{\boldsymbol{y}}_{\boldsymbol{PCA}}(s)$$

- Approach 2: Average class conditioned s-vectors (Avg s-vectors)

$$\hat{\boldsymbol{y}}_{\boldsymbol{avg}}(s) = \frac{1}{L}\sum_{l} \hat{\boldsymbol{y}}_l(s)$$

- Approach 3: Minimum mean squared error s-vectors (MMSE s-vectors)

$$\hat{\boldsymbol{y}}_{\boldsymbol{MMSE}}(s) = \sum_{l} p(l|X(s))\hat{\boldsymbol{y}}_l(s)$$

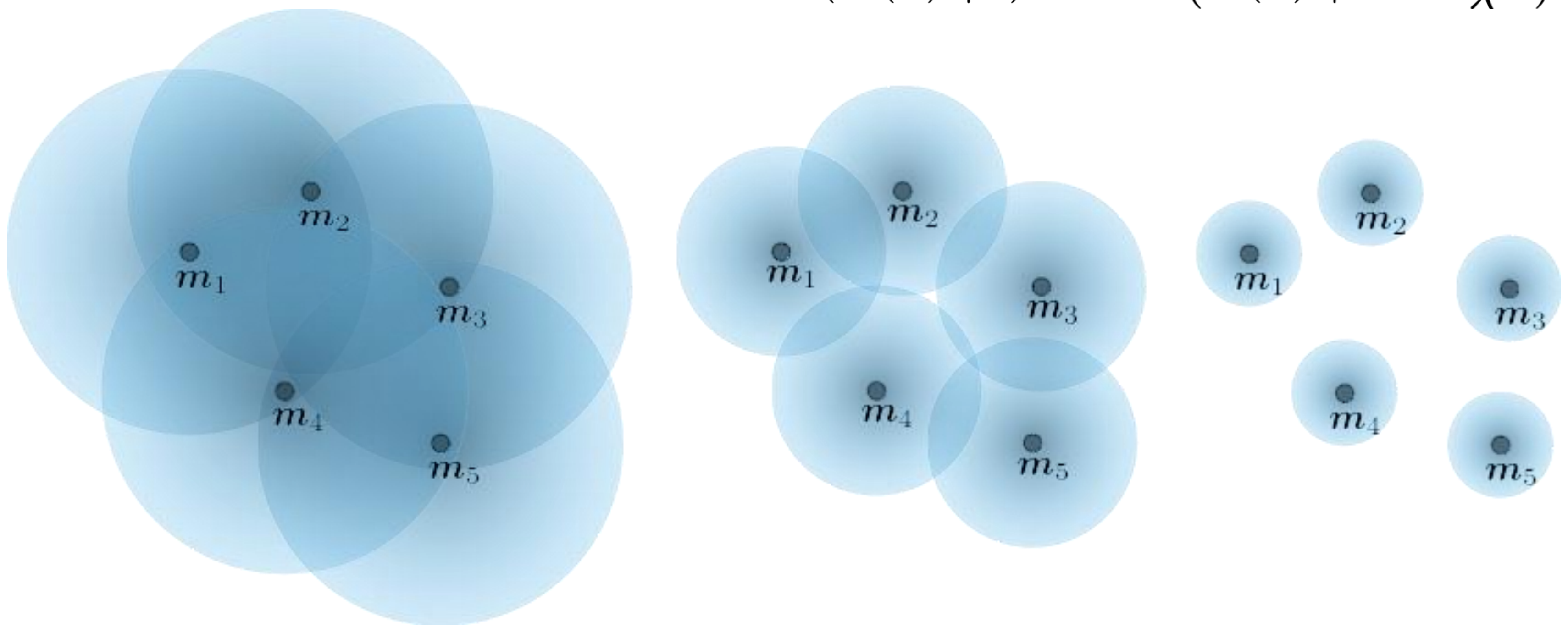
- Oracle s-vectors (cheat) : To understand the limits of this model.

$$\hat{\boldsymbol{y}}_{\boldsymbol{oracle}}(s) = \hat{\boldsymbol{y}}_{l(s)}(s)$$

Re-weighting the priors

- By re-weighting the prior covariance, the model can be made more discriminative

$$\mathrm{p}(\boldsymbol{y}(s) \mid l) = \mathcal{N}\left(\boldsymbol{y}(s) \mid \boldsymbol{m}_l, \tfrac{1}{\lambda} I\right)$$

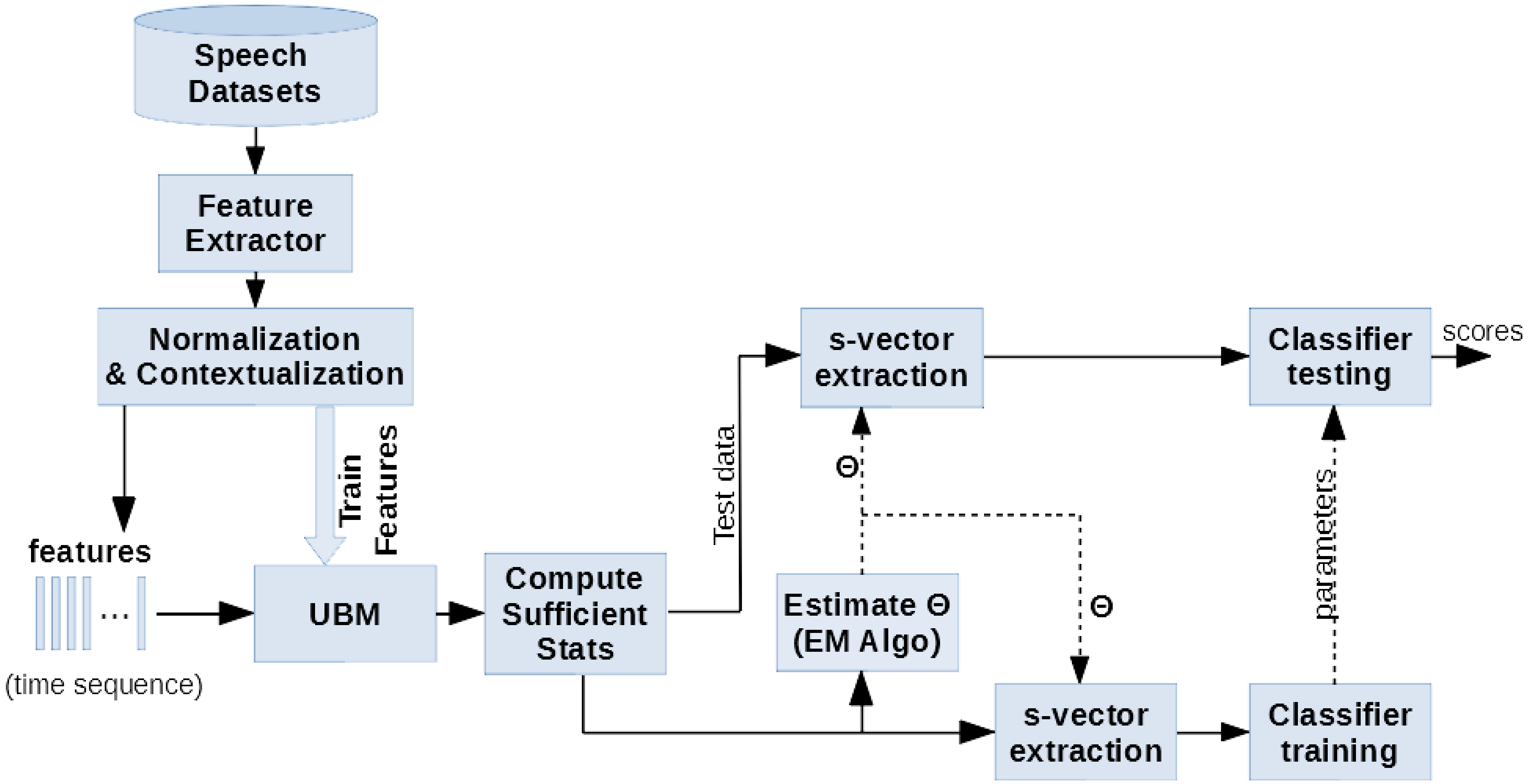


$\lambda = 1$ $\lambda = 3$ $\lambda = 5$

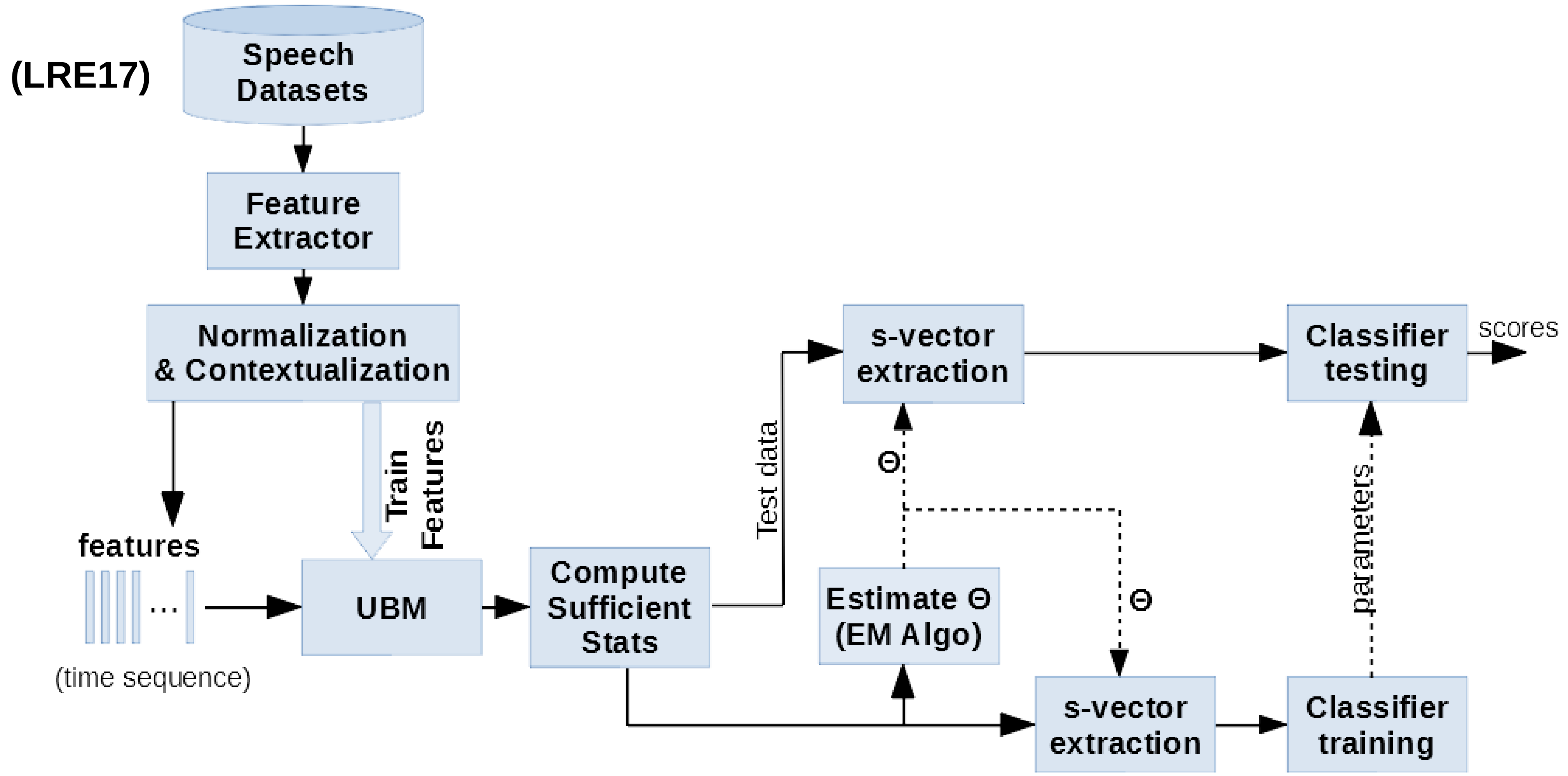
Experiments – NIST LRE 2017

Cluster	Target Languages	#files	Total Duration (hours)
Arabic	Egyptian Arabic (ara-arz)	440	190.9
	Iraqi Arabic (ara-acm)	1406	130.8
	Levantine Arabic (ara-apc)	3509	440.7
	Maghrebi Arabic (ara-ary)	919	81.8
Chinese	Mandarin (zho-cmn)	3331	379.4
	Min Nan (zho-nan)	95	13.3
English	British English (eng-gbr)	98	4.8
	General American English (eng-usg)	2448	327.7
Slavic	Polish (qsl-pol)	587	59.3
	Russian (qsl-rus)	1221	69.5
Iberian	Caribbean Spanish (spa-car)	688	166.3
	European Spanish (spa-eur)	121	24.7
	Latin American Spanish (spa-lac)	898	175.9
	Brazilian Portuguese (por-brz)	444	4.1

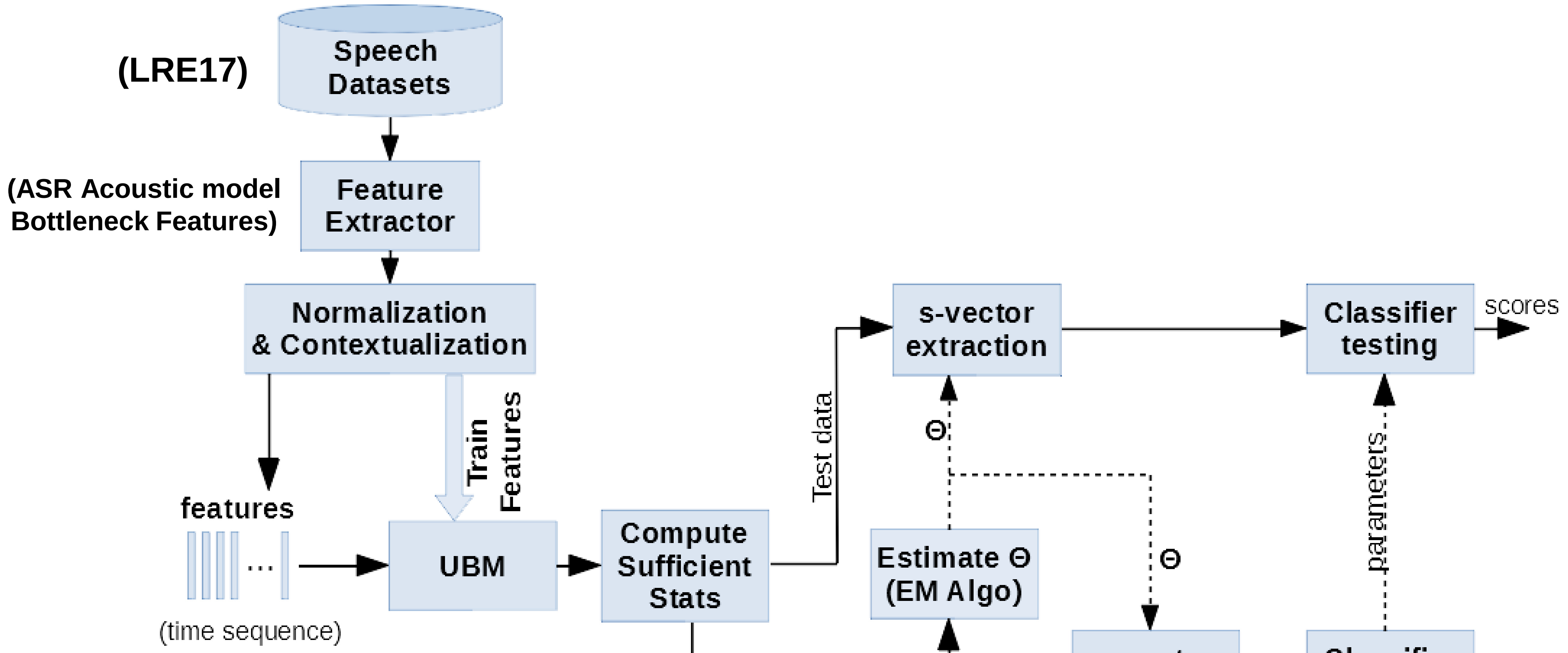
Experiments – NIST LRE 2017



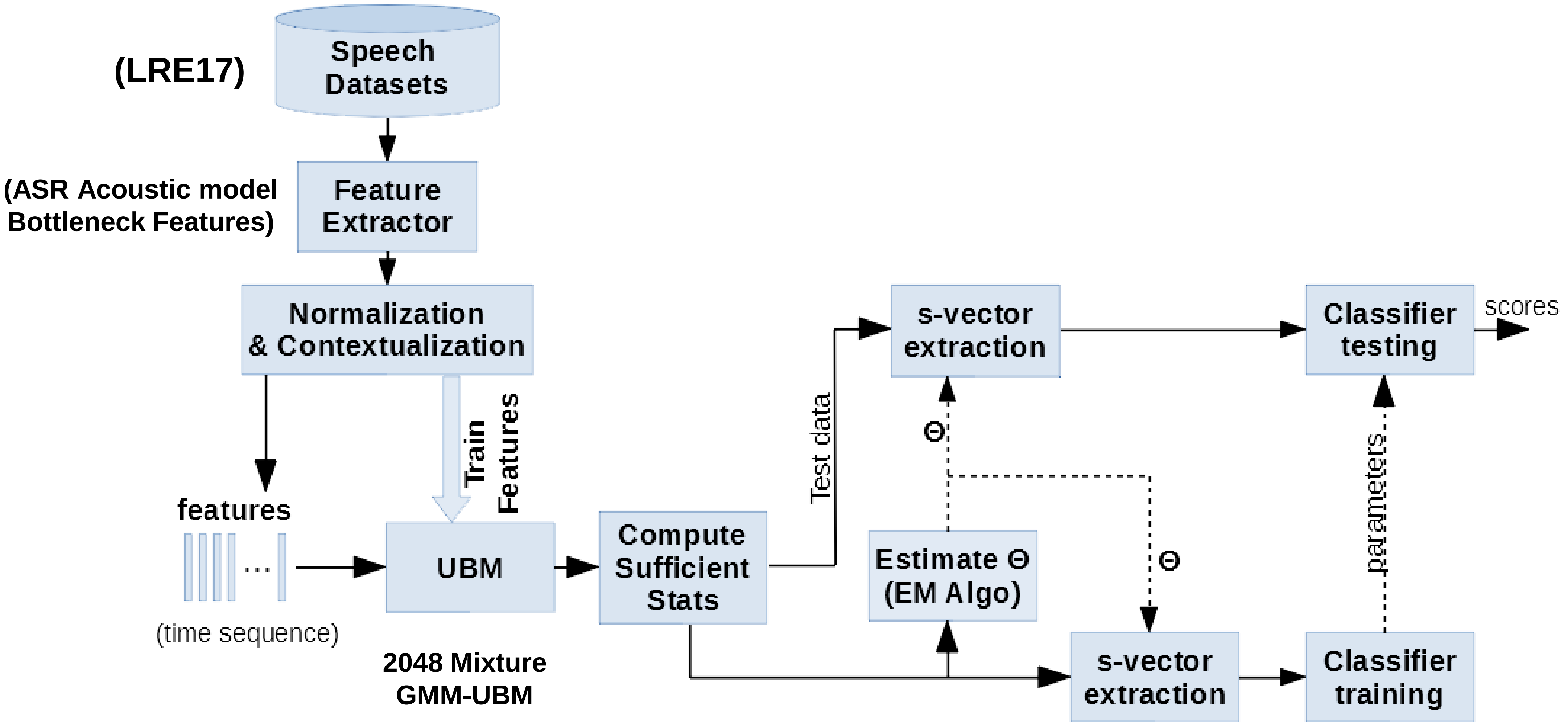
Experiments – NIST LRE 2017



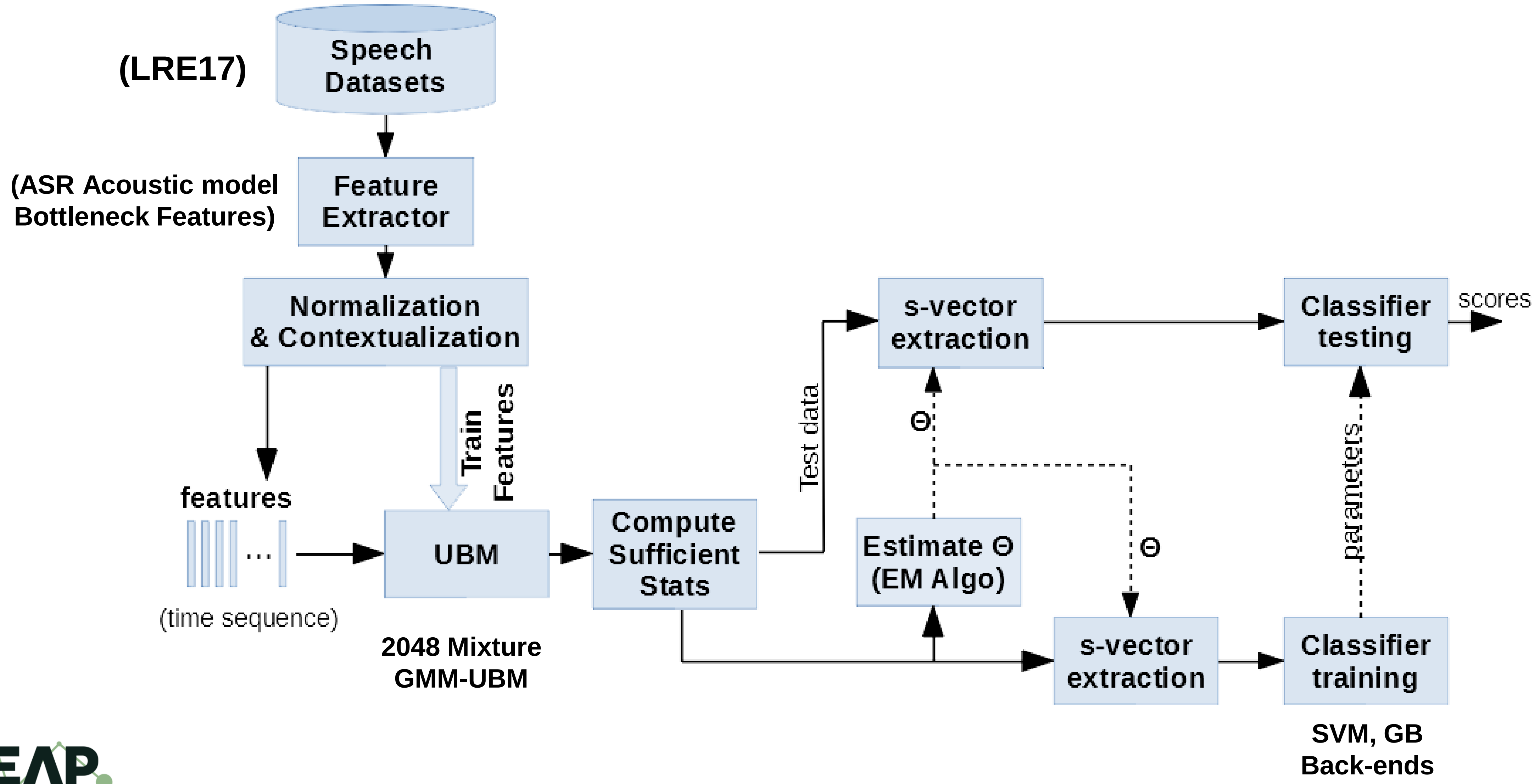
Experiments – NIST LRE 2017



Experiments – NIST LRE 2017



Experiments – NIST LRE 2017



Results

- Results on LRE17 Development Dataset (3661 test examples) with SVM Back-end.

	Dev Performances : $100C_{primary}$ [EER (%)] {Accuracy(%)}		
Model config.	3 sec	10 sec	30 sec
Unsupervised i-vector [1]	52.7 [16.6] {51.8}	27.1 [7.5] {74.0}	13.1 [3.6] {87.8}
Sup. i-vector [2]	47.2 [15.1] {61.1}	20.2 [5.7] {80.9}	14.8 [4.1] {85.1}
Simplified Sup. i-vector [2]	58.2 [19.6] {47.8}	27.3 [7.8] {73.6}	13.4 [3.7] {87.7}
LSTM [3]	53.7 [15.39] {52.7}	33.8 [9.7] {69.2}	32.0 [8.81] {70.9}
HGRU [4]	53.1 [15.09] {57.5}	27.6 [6.6] {76.9}	25.5 [6.1] {78.6}
Average s-vector	47.8 [14.3] {56.3}	22.4 [6.2] {78.3}	12.2 [**3.3**] {88.0}
PCA s-vector	57.6 [18.2] {50.4}	27.0 [7.4] {74.6}	12.1 [3.4] {88.2}
Approx. MMSE s-vector	51.1 [15.3] {54.1}	23.9 [6.1] {77.4}	12.1 [3.3] {88.3}
MMSE s-vector	**44.4** [**14.7**] {**63.5**}	**19.5** [**5.9**] {**83.7**}	**11.7** [3.4] {**89.4**}
Oracle s-vector (cheat)	13.0 [3.9] {88.3}	5.6 [1.7] {94.4}	5.0 [1.1] {94.9}

[1] Dehak et. al., "*Language Recognition via i-vectors and dimensionality reduction*", Interspeech 2011
[2] M. Li et. al., "Simplified supervised i-vector modeling …" CSL 2014
[3] R. Zazo et. al., "Evaluation of an LSTM RNN system …" Odyssey 2016
[4] B. Padi et. al., "End-to-end language recognition using attention based HGRUs" ICASSP 2019

Results

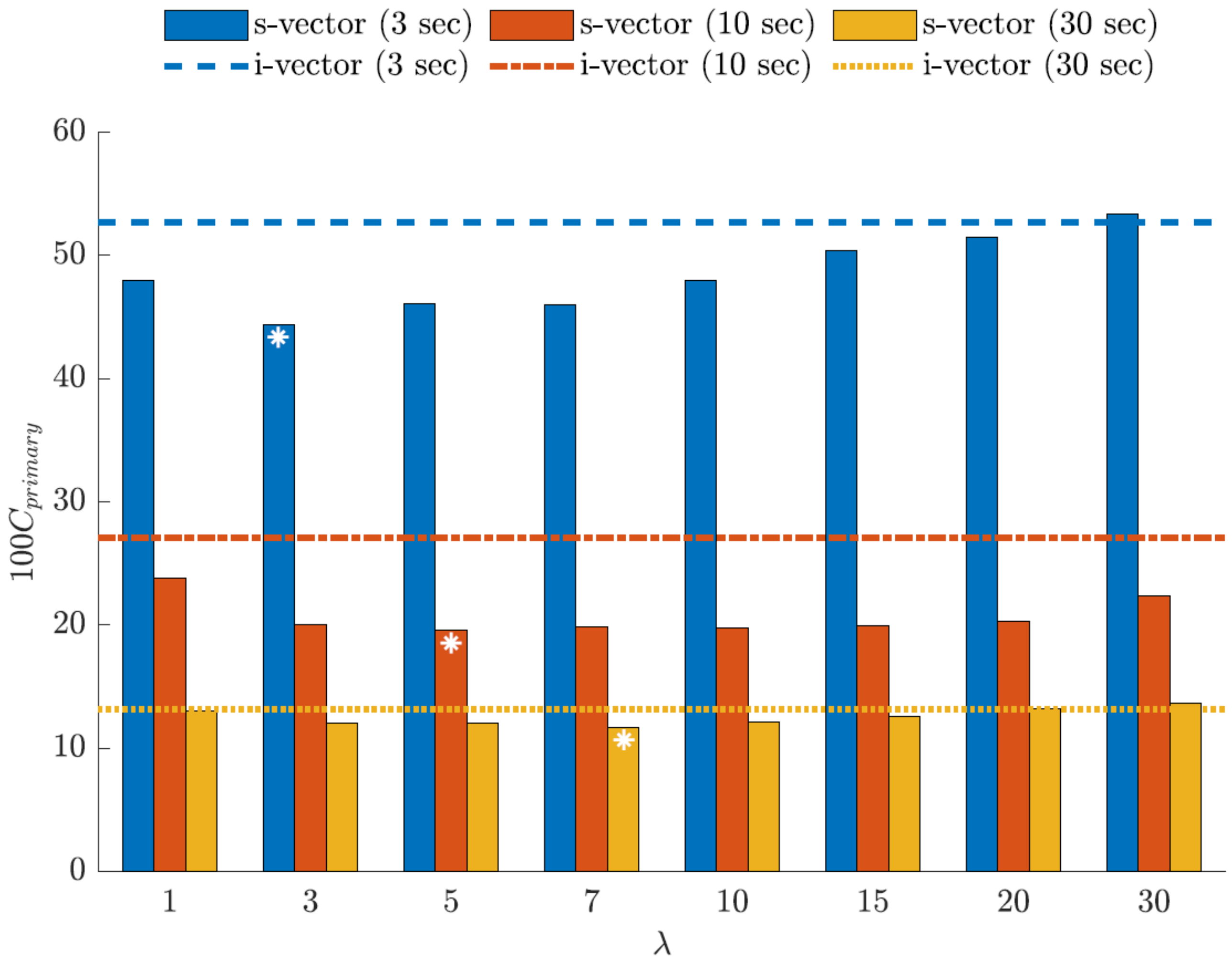
- Results on LRE17 Evaluation Dataset (25k test examples) with SVM Back-end.

	Eval Performances : $100C_{primary}$ [EER (%)] {Accuracy(%)}		
Model config.	3 sec	10 sec	30 sec
Unsupervised i-vector [1]	53.6 [16.1] {53.8}	29.9 [8.6] {72.4}	16.7 [3.9] {83.0}
Sup. i-vector [2]	46.1 [14.7] {59.6}	25.6 [7.3] {76.4}	19.9 [5.1] {80.9}
Simplified Sup. i-vector [2]	57.2 [19.1] {49.8}	29.8 [8.4] {71.4}	16.7 [4.1] {82.8}
LSTM [3]	55.2 [15.4] {54.7}	35.4 [8.7] {72.1}	28.1 [7.3] {76.1}
HGRU [4]	55.4 [15.3] {55.1}	32.3 [7.5] {74.1}	23.3 [4.9] {83.0}
Average s-vector	49.7 [13.9] {58.5}	27.0 [7.0] {75.3}	15.7 [**3.7**] {84.0}
PCA s-vector	58.2 [17.7] {54.1}	30.5 [8.2] {73.7}	15.8 [3.8] {83.9}
Approx. MMSE s-vector	52.2 [14.6] {56.8}	27.8 [7.1] {74.7}	15.8 [3.6] {84.0}
MMSE s-vector	**43.7** [**13.5**] {**61.2**}	**23.7** [**6.5**] {**77.4**}	**15.4** [3.8] {**84.5**}
Oracle s-vector (cheat)	12.6 [3.6] {86.9}	6.5 [1.6] {92.9}	5.7 [1.4] {93.6}

[1] Dehak et. al., “*Language Recognition via i-vectors and dimensionality reduction*”, Interspeech 2011
[2] M. Li et. al., “*Simplified supervised i-vector modeling …*” CSL 2014
[3] R. Zazo et. al., “*Evaluation of an LSTM RNN system …*” Odyssey 2016
[4] B. Padi et. al., “*End-to-end language recognition using attention based HGRUs*” ICASSP 2019

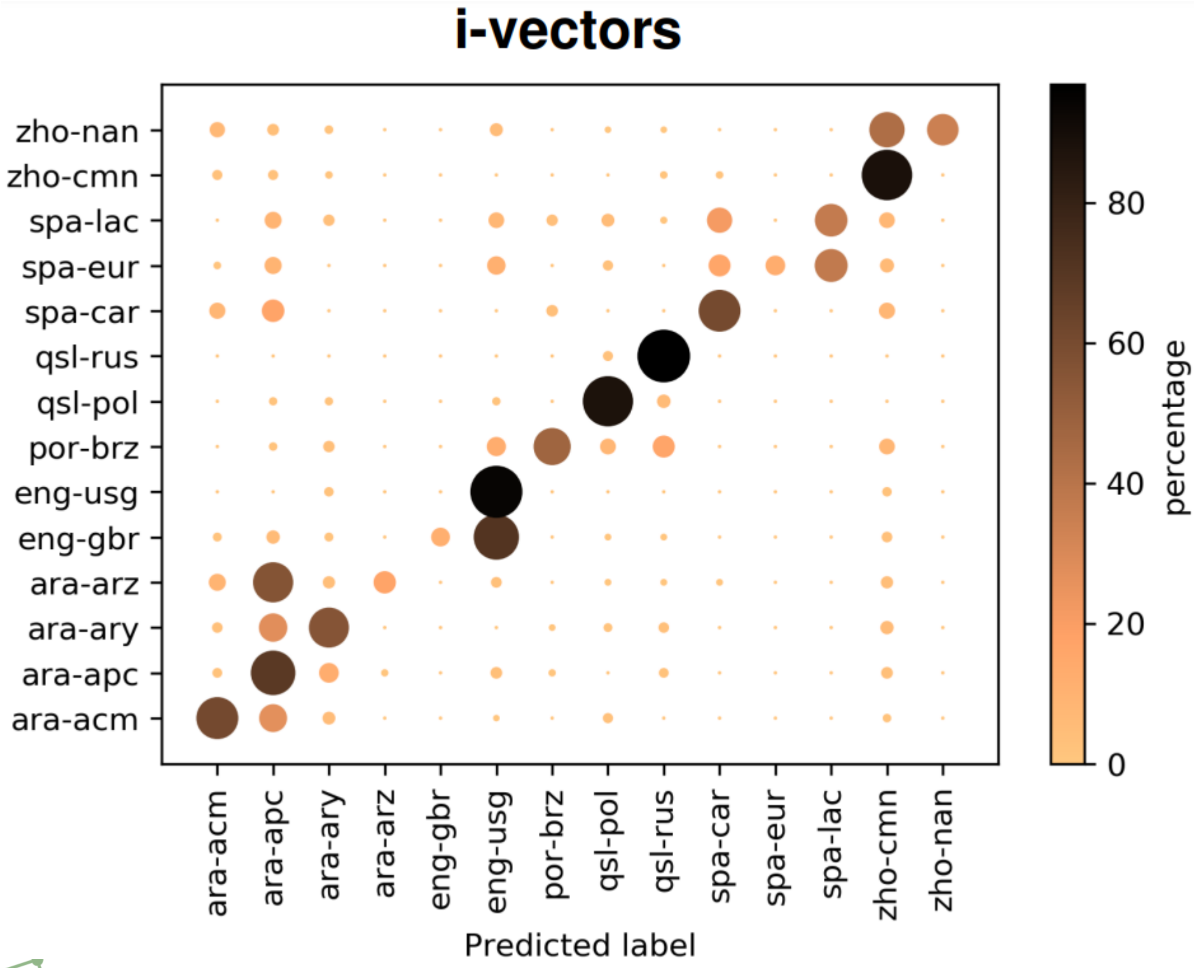
Results

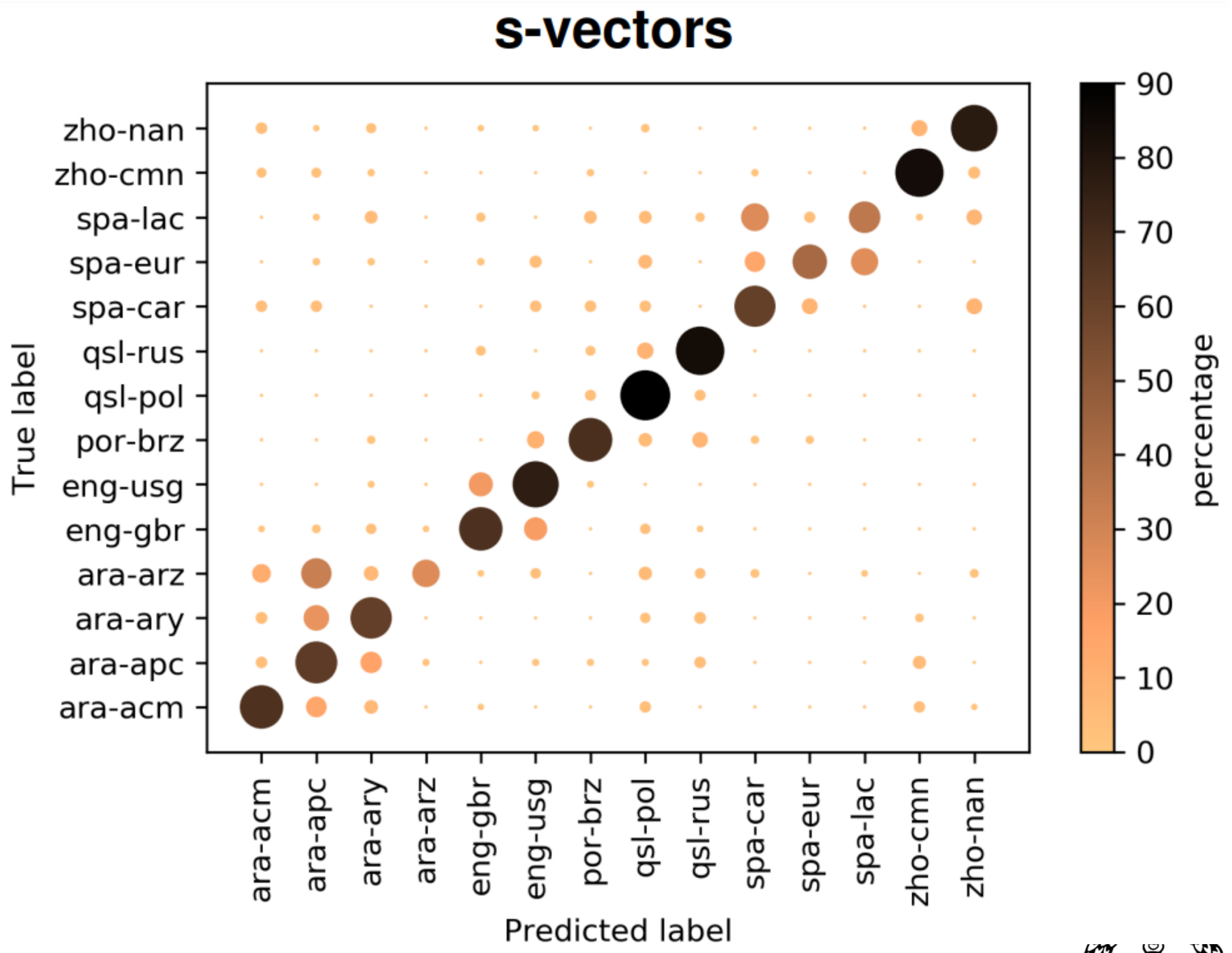
- Variation of $C_{Primary}$ with λ for various durations for MMSE s-vectors.



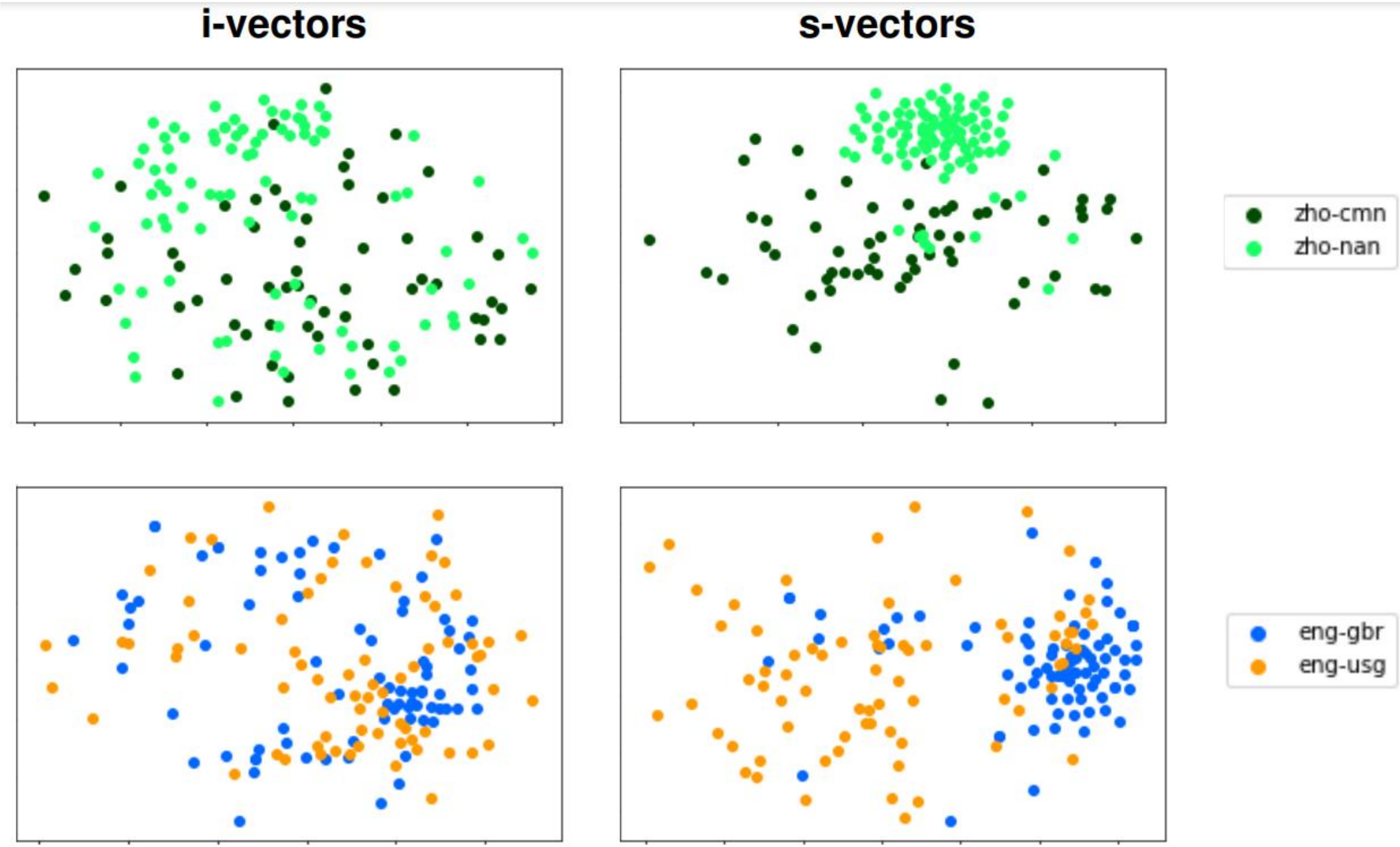
Results

- Comparison of confusion matrices for 3 sec condition with LRE2017 dev set.





t-SNE Visualization



Summary of s-vector modeling

- We proposed a supervised approach to i-vector modeling by assuming a GMM prior on the latent variables in the i-vector model.

Summary of s-vector modeling

- We proposed a supervised approach to i-vector modeling by assuming a GMM prior on the latent variables in the i-vector model.

- We derived the likelihood expressions for the E-M Algorithm used in parameter estimation.

Summary of s-vector modeling

- We proposed a supervised approach to i-vector modeling by assuming a GMM prior on the latent variables in the i-vector model.
- We derived the likelihood expressions for the E-M Algorithm used in parameter estimation.
- Re-weighting the priors for more discriminative s-vectors.

Summary of s-vector modeling

- We proposed a supervised approach to i-vector modeling by assuming a GMM prior on the latent variables in the i-vector model.
- We derived the likelihood expressions for the E-M Algorithm used in parameter estimation.
- Re-weighting the priors for more discriminative s-vectors.
- We observed consistent improvements for all the durations (3 sec, 10 sec, 30 sec) conditions with s-vectors, over the i-vector baseline.

Summary of s-vector modeling

- We proposed a supervised approach to i-vector modeling by assuming a GMM prior on the latent variables in the i-vector model.

- We derived the likelihood expressions for the E-M Algorithm used in parameter estimation.

- Re-weighting the priors for more discriminative s-vectors.

- We observed consistent improvements for all the durations (3 sec, 10 sec, 30 sec) conditions with s-vectors, over the i-vector baseline.

- Confusion matrices, data visualization using t-SNE embeddings show that s-vectors are highly useful for accent recognition.

Summary of s-vector modeling

- We proposed a supervised approach to i-vector modeling by assuming a GMM prior on the latent variables in the i-vector model.
- We derived the likelihood expressions for the E-M Algorithm used in parameter estimation.
- Re-weighting the priors for more discriminative s-vectors.
- We observed consistent improvements for all the durations (3 sec, 10 sec, 30 sec) conditions with s-vectors, over the i-vector baseline.
- Confusion matrices, data visualization using t-SNE embeddings show that s-vectors are highly useful for accent recognition.
- Downside: Memory/time requirement increases linearly with number of class labels used.

Supervised Neural Network Models for Speaker Verification

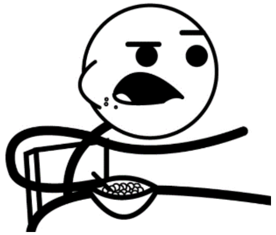
Speaker Verification

- Given an “enrollment recording” (sample speech audio) of a particular (target) speaker of interest, determine whether a “test audio segment” is spoken by (or contains) the target speaker or not.

Speaker Verification

- Given an “enrollment recording” (sample speech audio) of a particular (target) speaker of interest, determine whether a “test audio segment” is spoken by (or contains) the target speaker or not.

***target speaker**

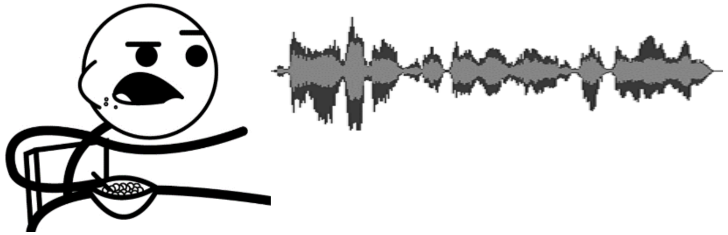


Speaker Verification

- Given an “enrollment recording” (sample speech audio) of a particular (target) speaker of interest, determine whether a “test audio segment” is spoken by (or contains) the target speaker or not.

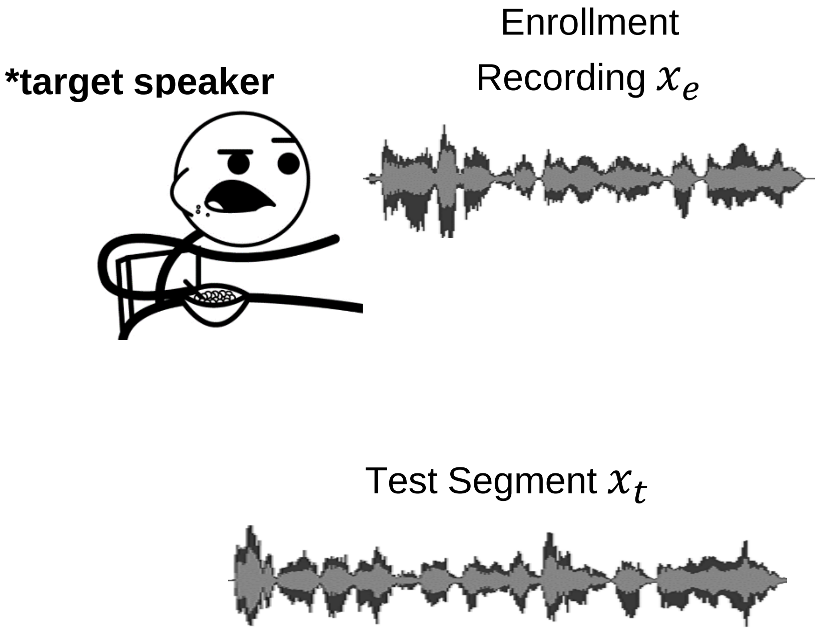
***target speaker**

Enrollment Recording x_e



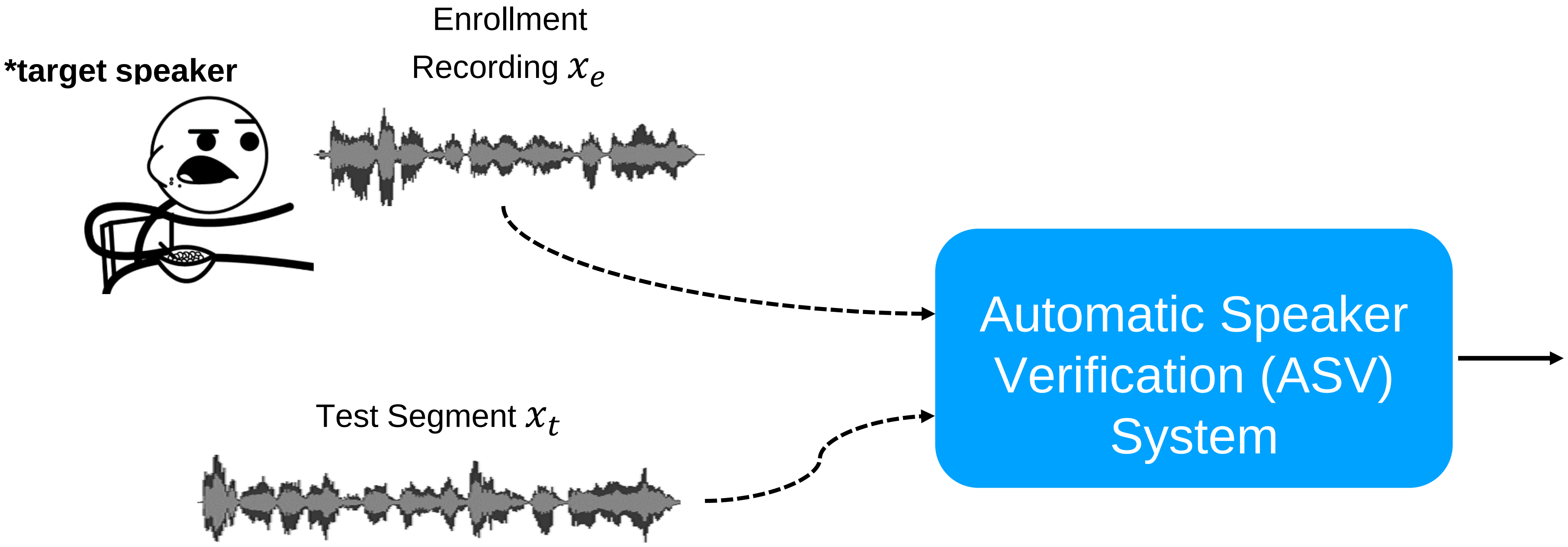
Speaker Verification

- Given an "enrollment recording" (sample speech audio) of a particular (target) speaker of interest, determine whether a "test audio segment" is spoken by (or contains) the target speaker or not.



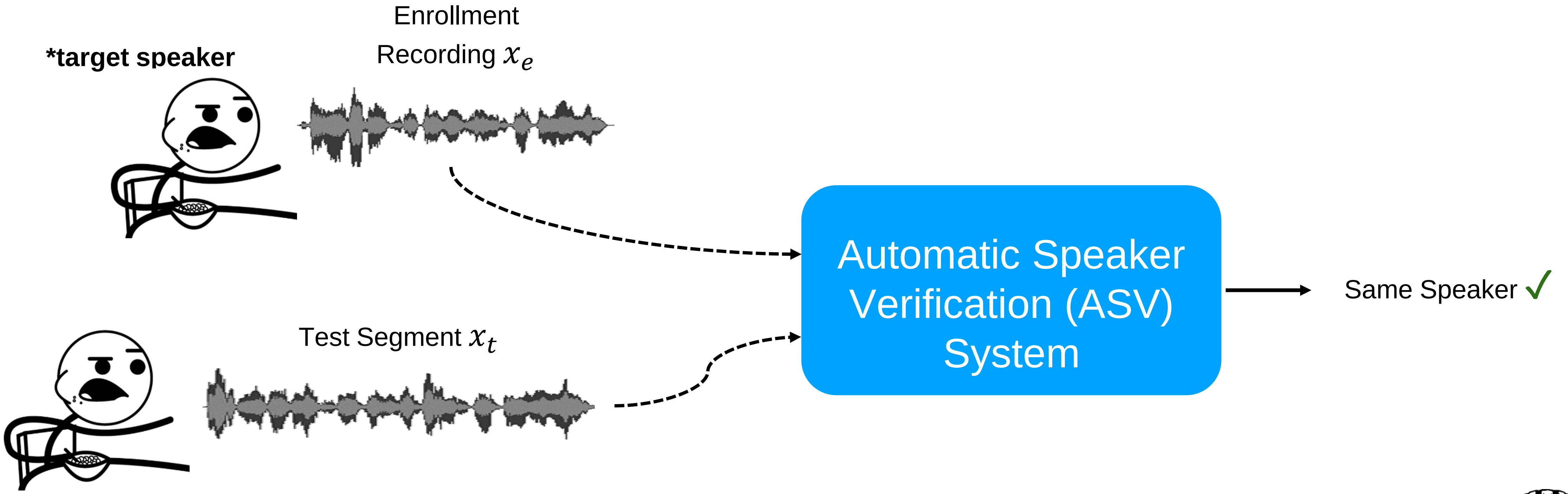
Speaker Verification

- Given an “enrollment recording” (sample speech audio) of a particular (target) speaker of interest, determine whether a “test audio segment” is spoken by (or contains) the target speaker or not.



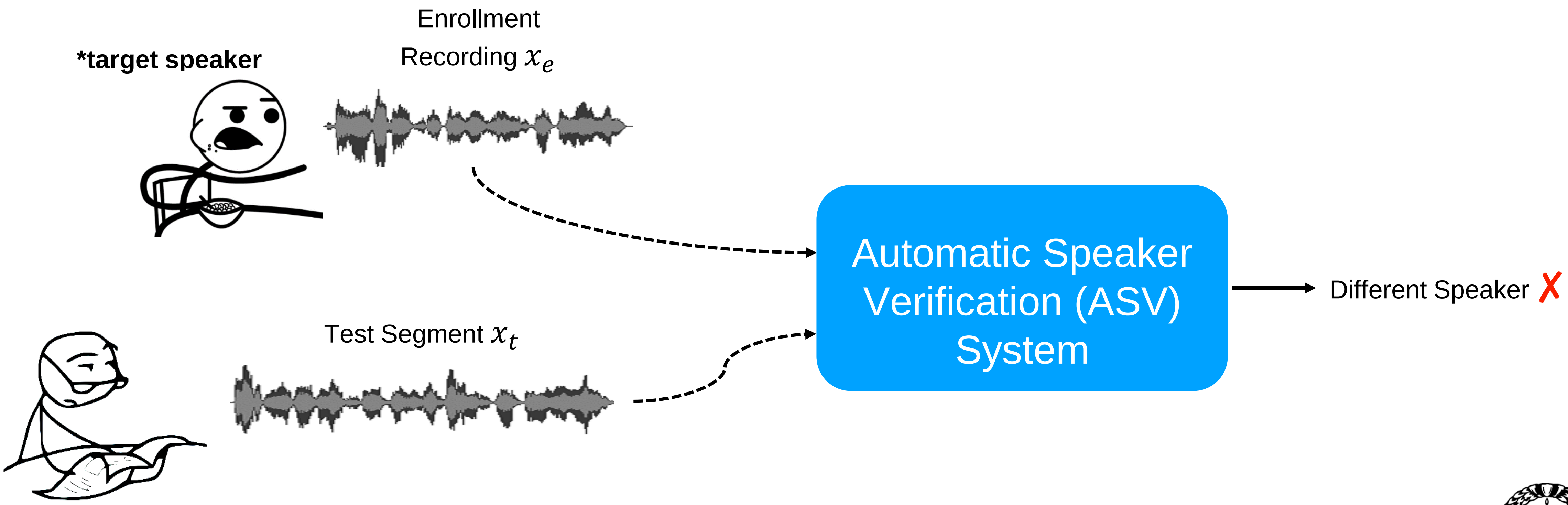
Speaker Verification

- Given an “enrollment recording” (sample speech audio) of a particular (target) speaker of interest, determine whether a “test audio segment” is spoken by (or contains) the target speaker or not.



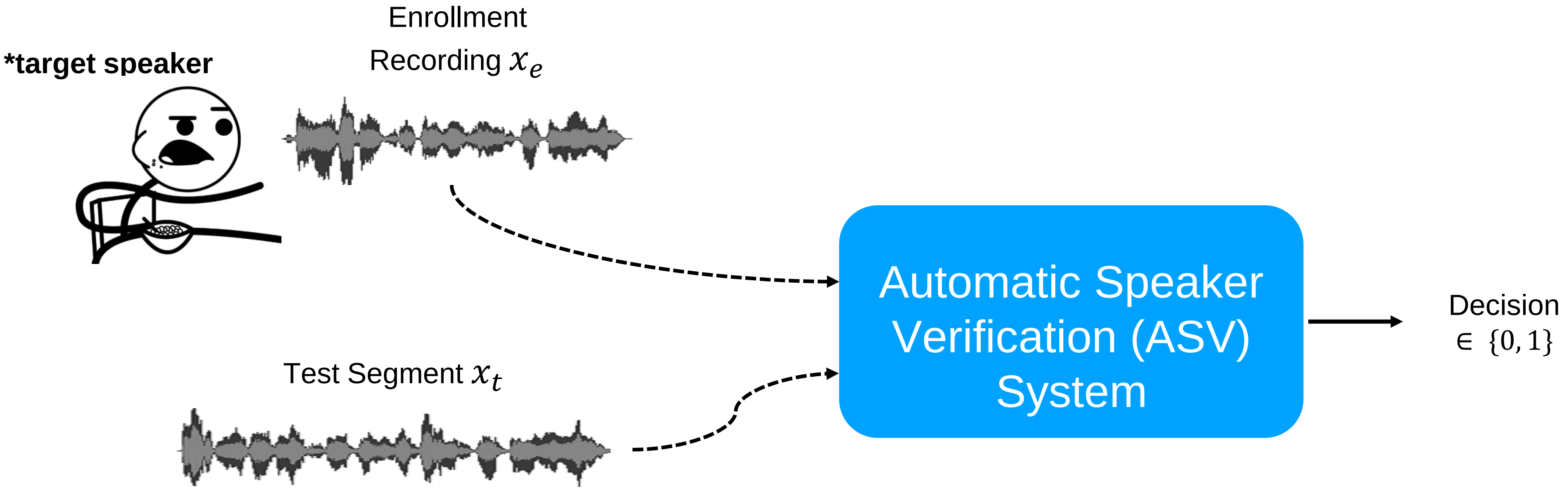
Speaker Verification

- Given an “enrollment recording” (sample speech audio) of a particular (target) speaker of interest, determine whether a “test audio segment” is spoken by (or contains) the target speaker or not.



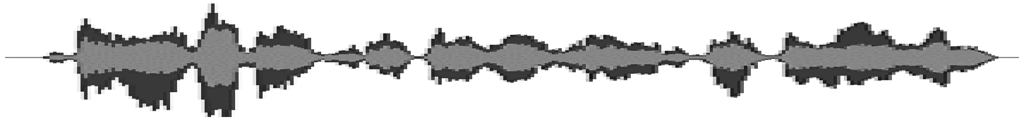
Speaker Verification

- Given an “enrollment recording” (sample speech audio) of a particular (target) speaker of interest, determine whether a “test audio segment” is spoken by (or contains) the target speaker or not.

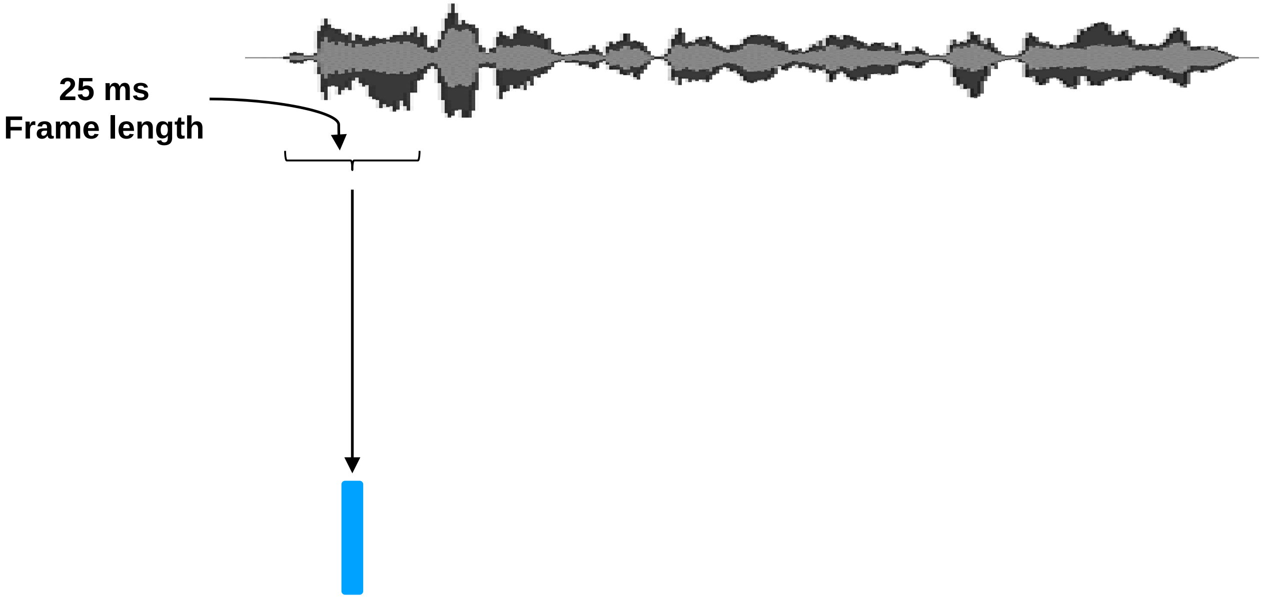


General Pipeline of NN based ASV

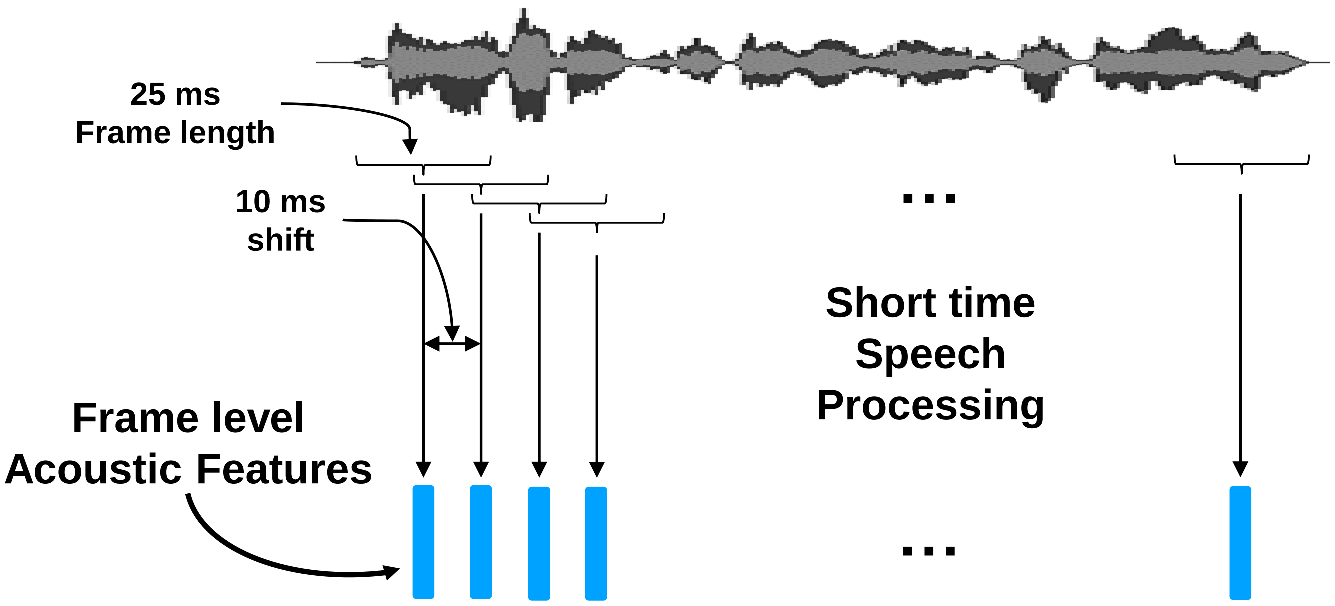
General Pipeline of NN based ASV



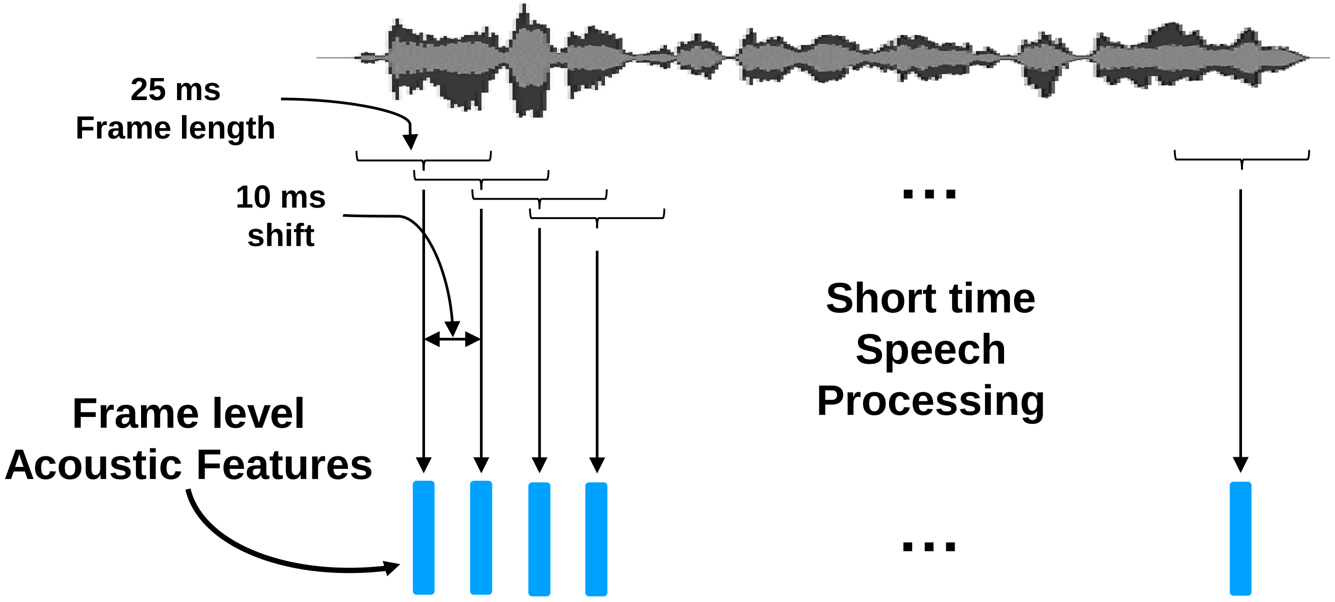
General Pipeline of NN based ASV



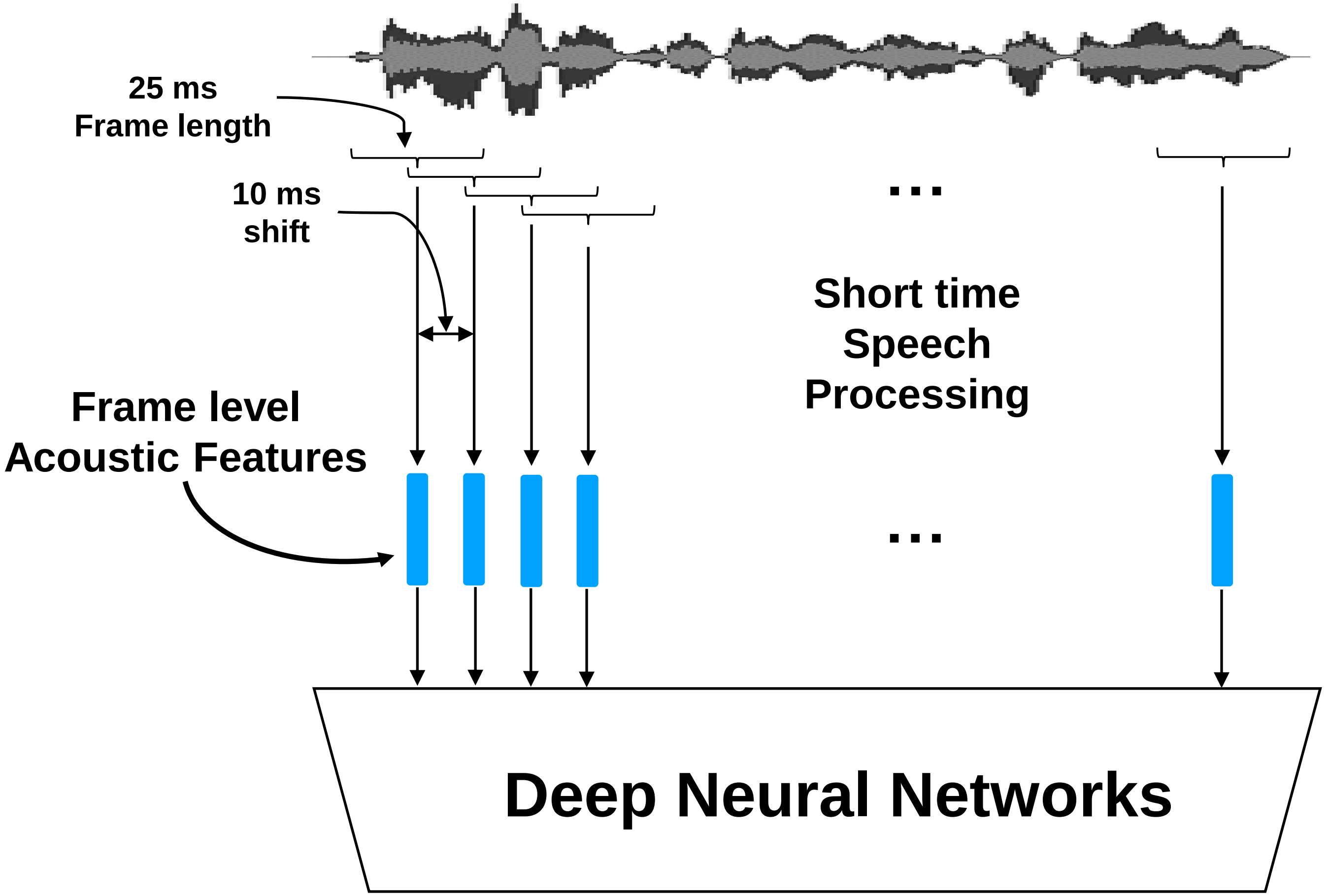
General Pipeline of NN based ASV



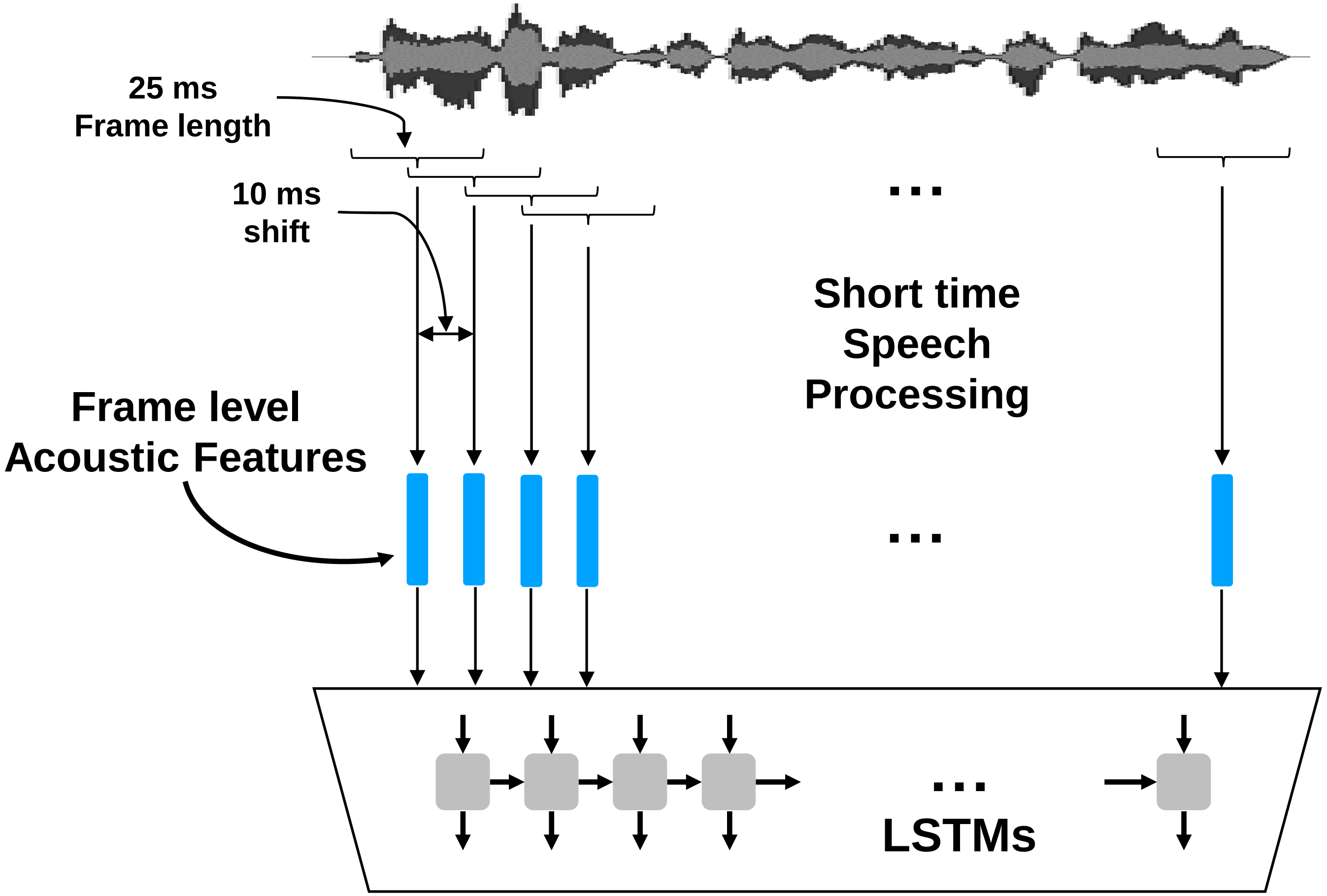
General Pipeline of NN based ASV



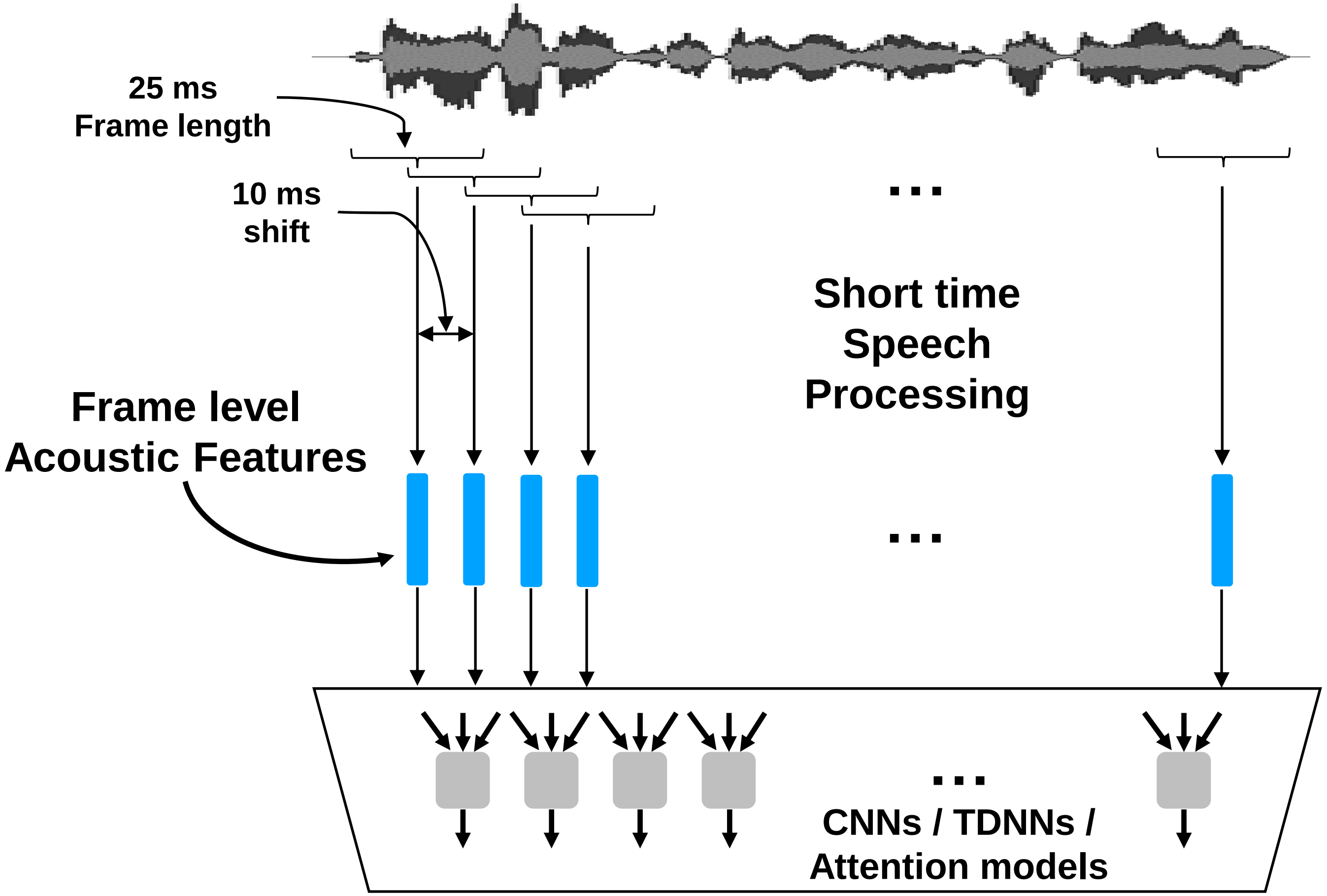
General Pipeline of NN based ASV



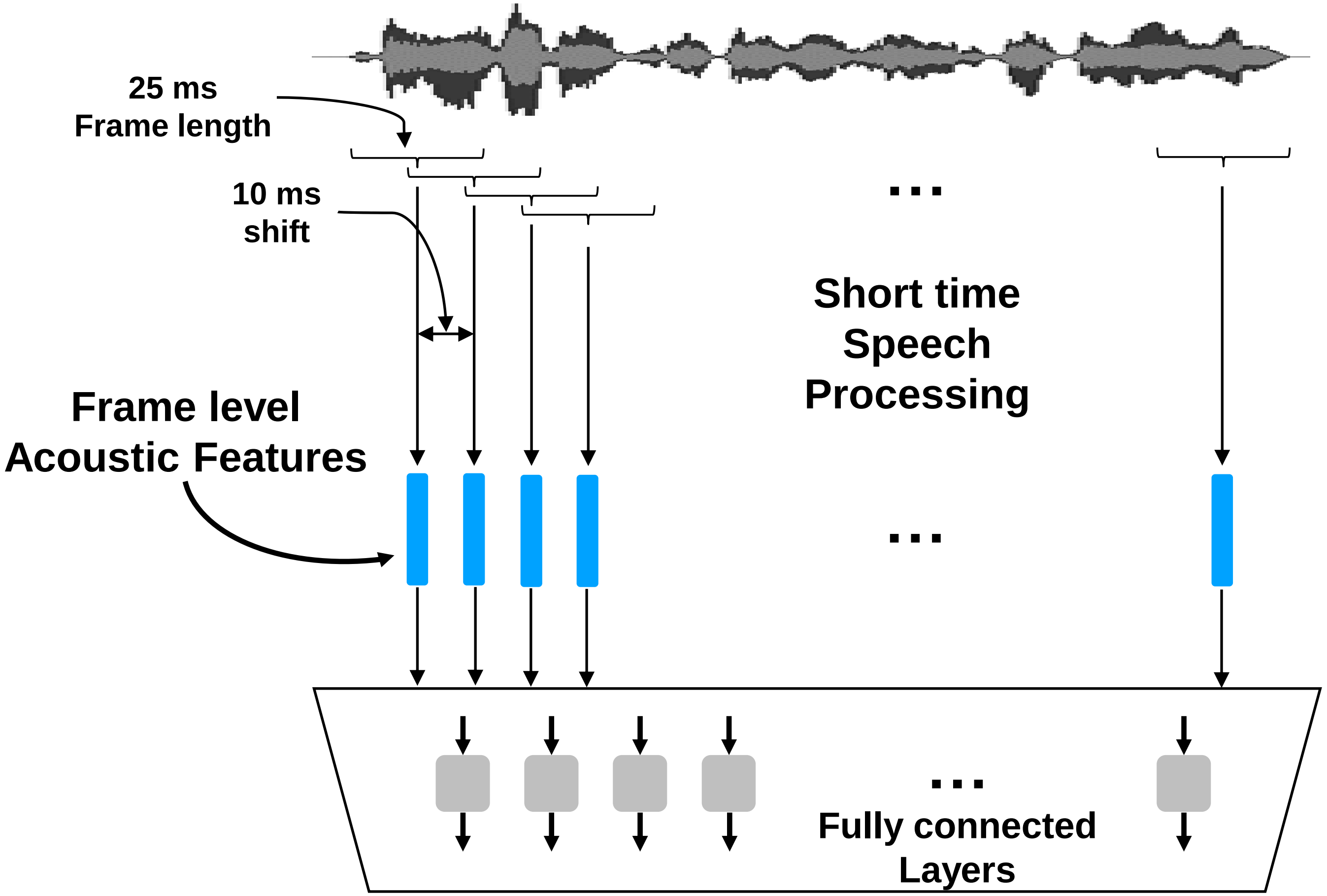
General Pipeline of NN based ASV



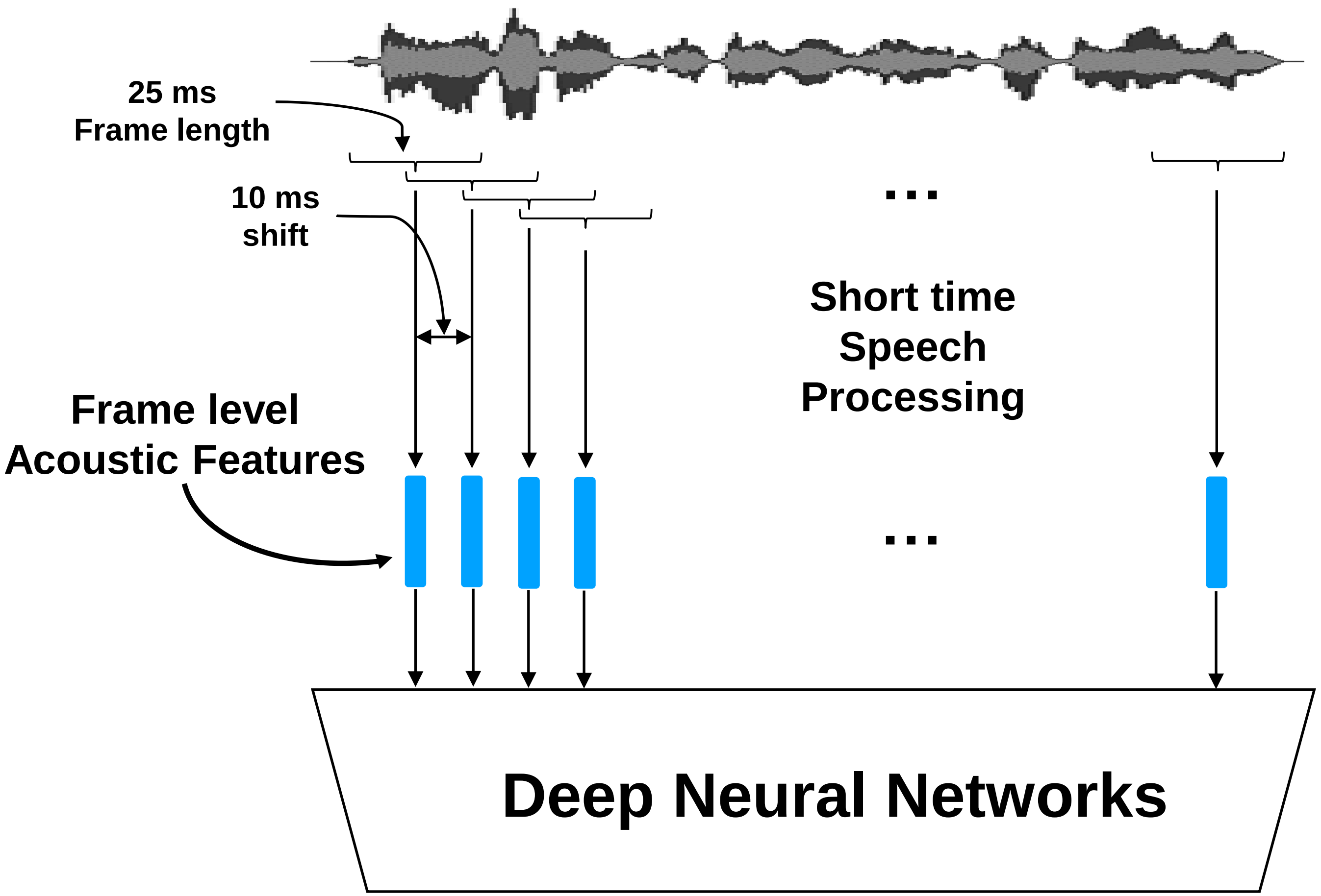
General Pipeline of NN based ASV



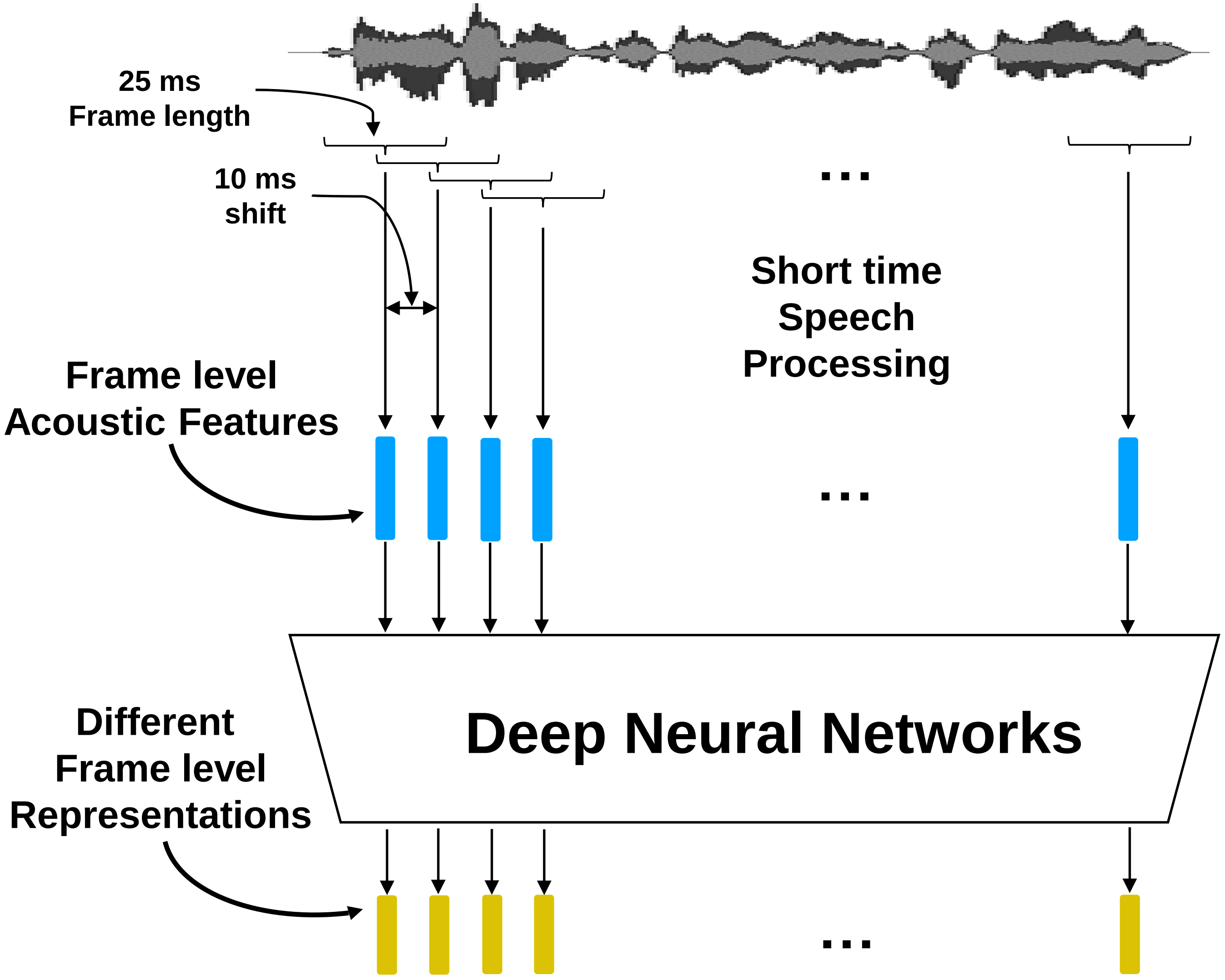
General Pipeline of NN based ASV



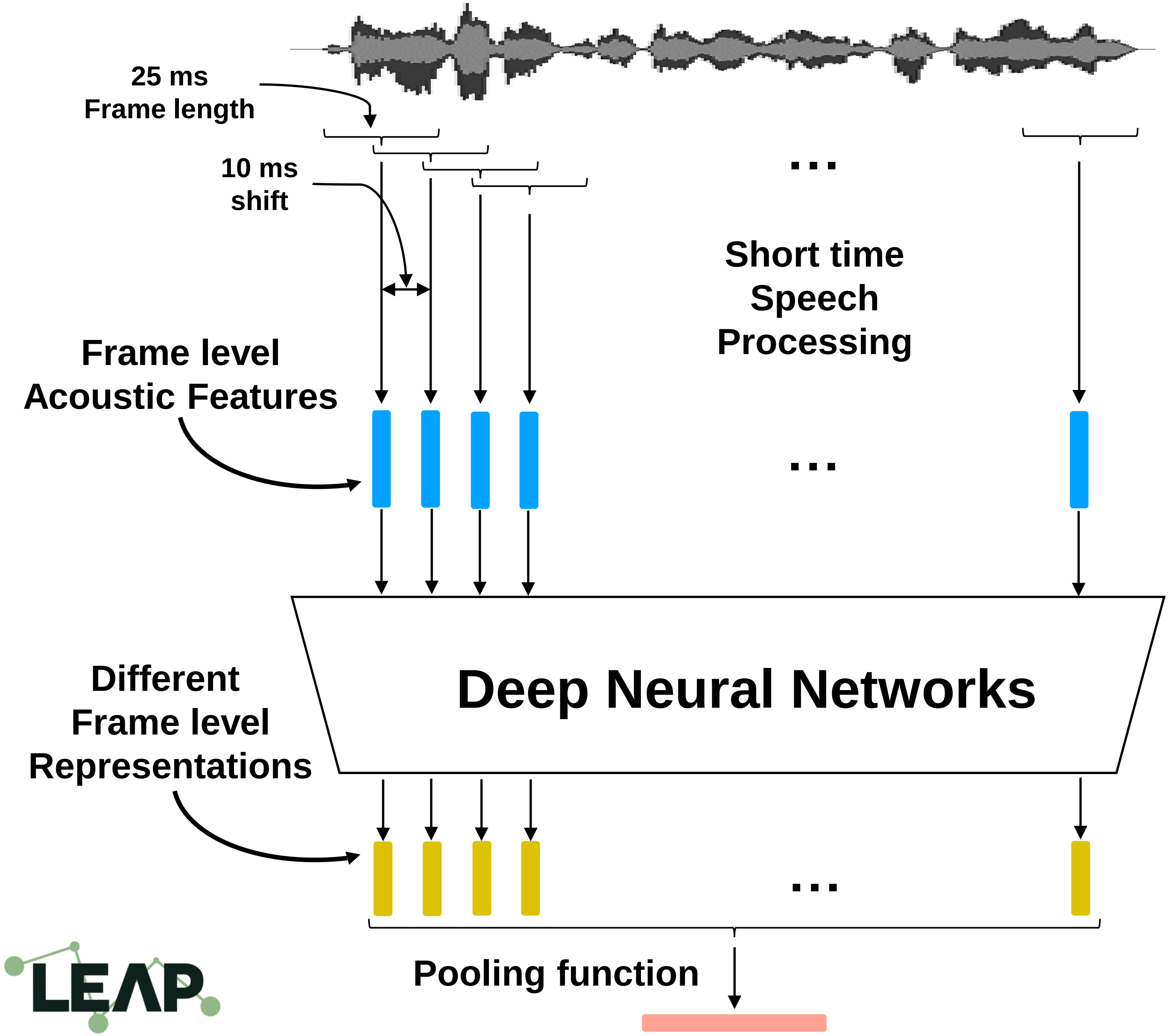
General Pipeline of NN based ASV



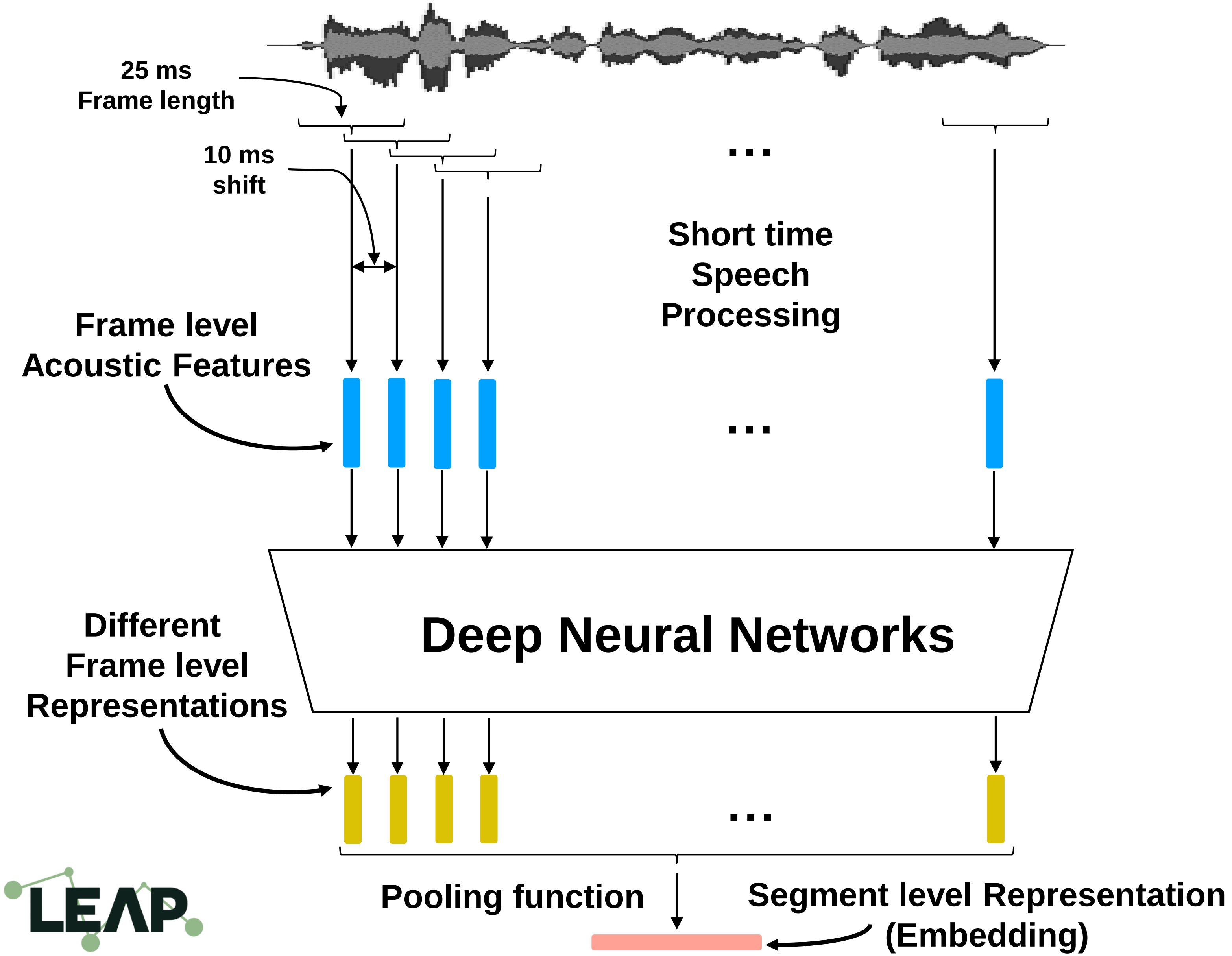
General Pipeline of NN based ASV



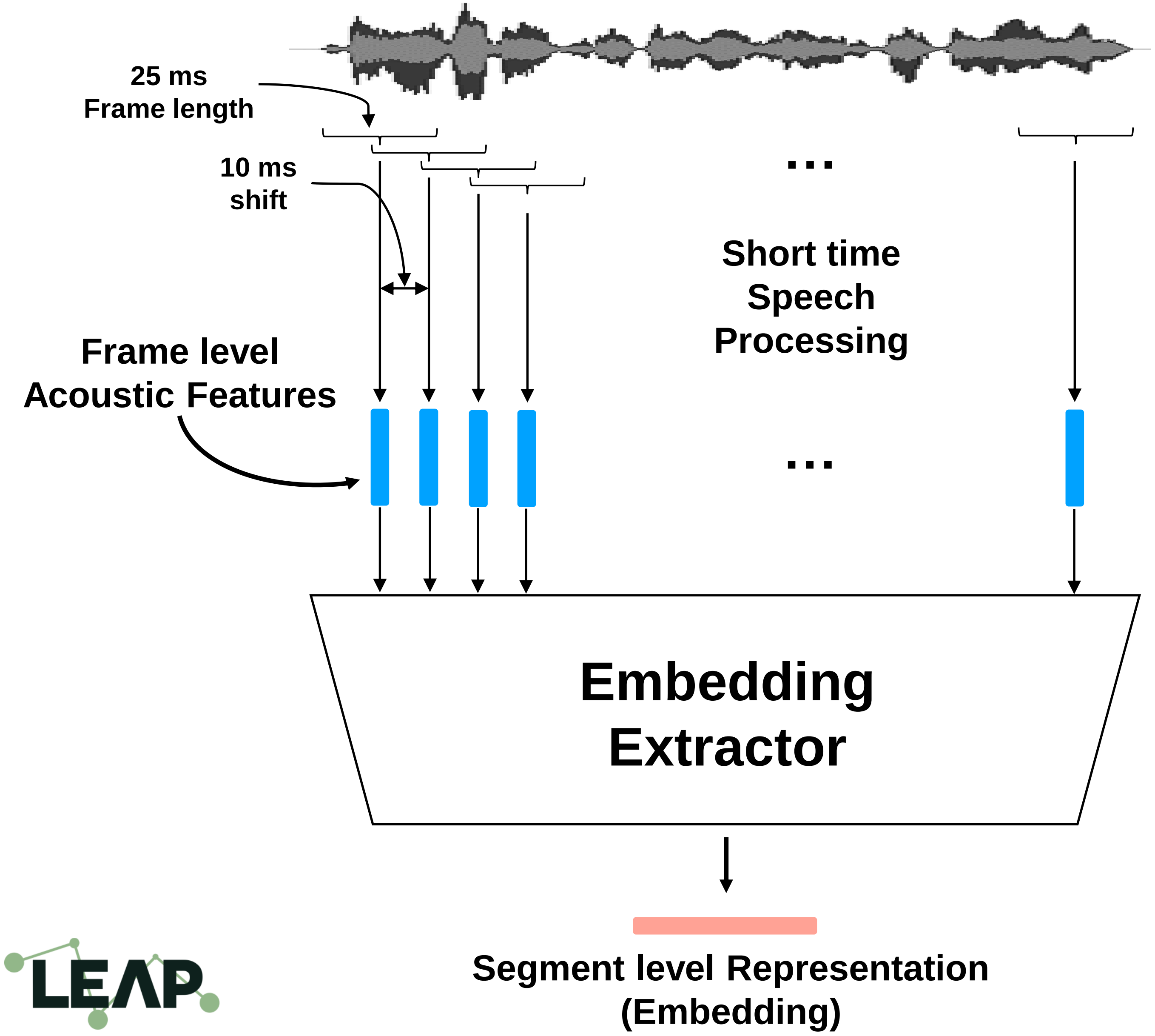
General Pipeline of NN based ASV



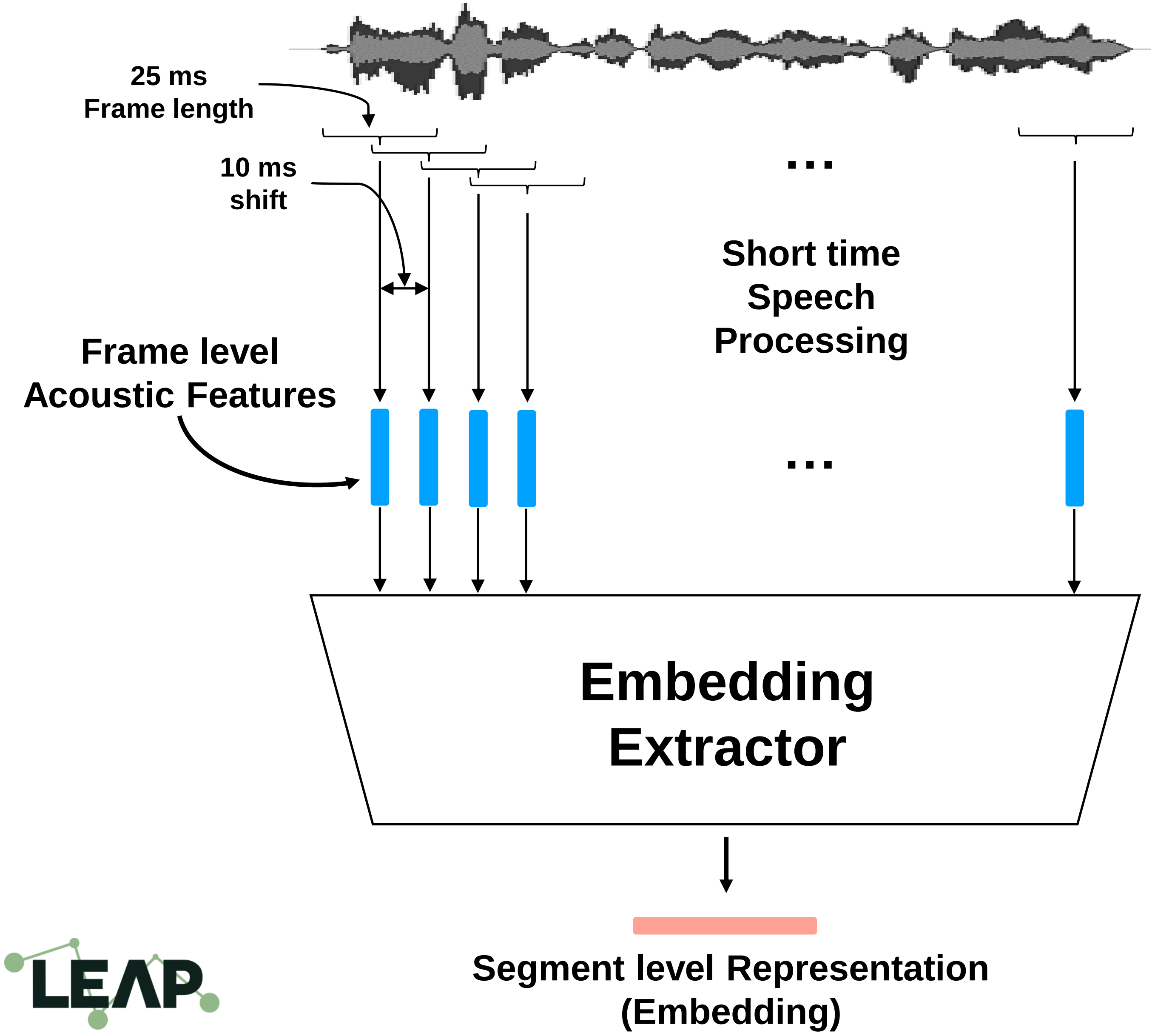
General Pipeline of NN based ASV



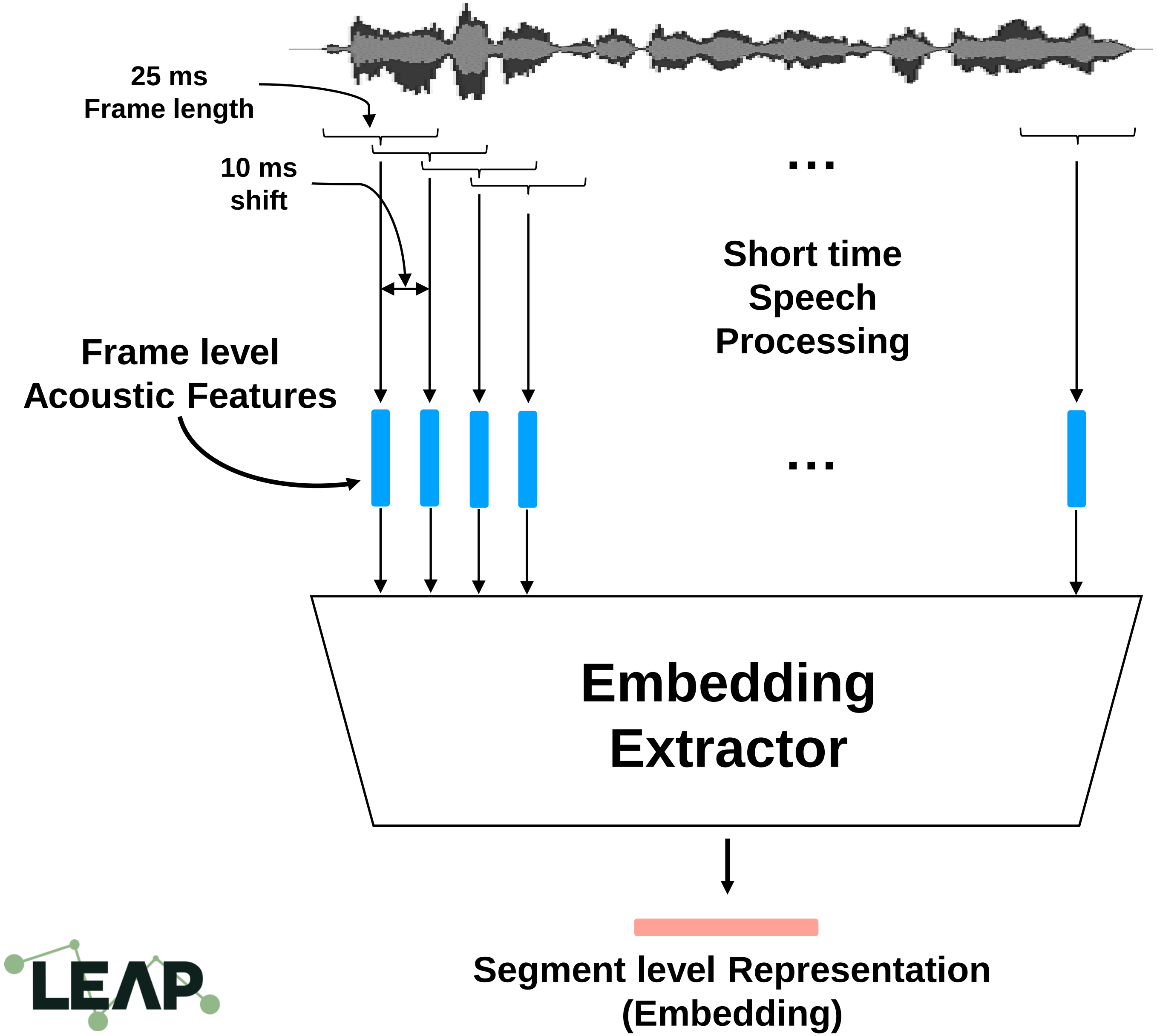
General Pipeline of NN based ASV



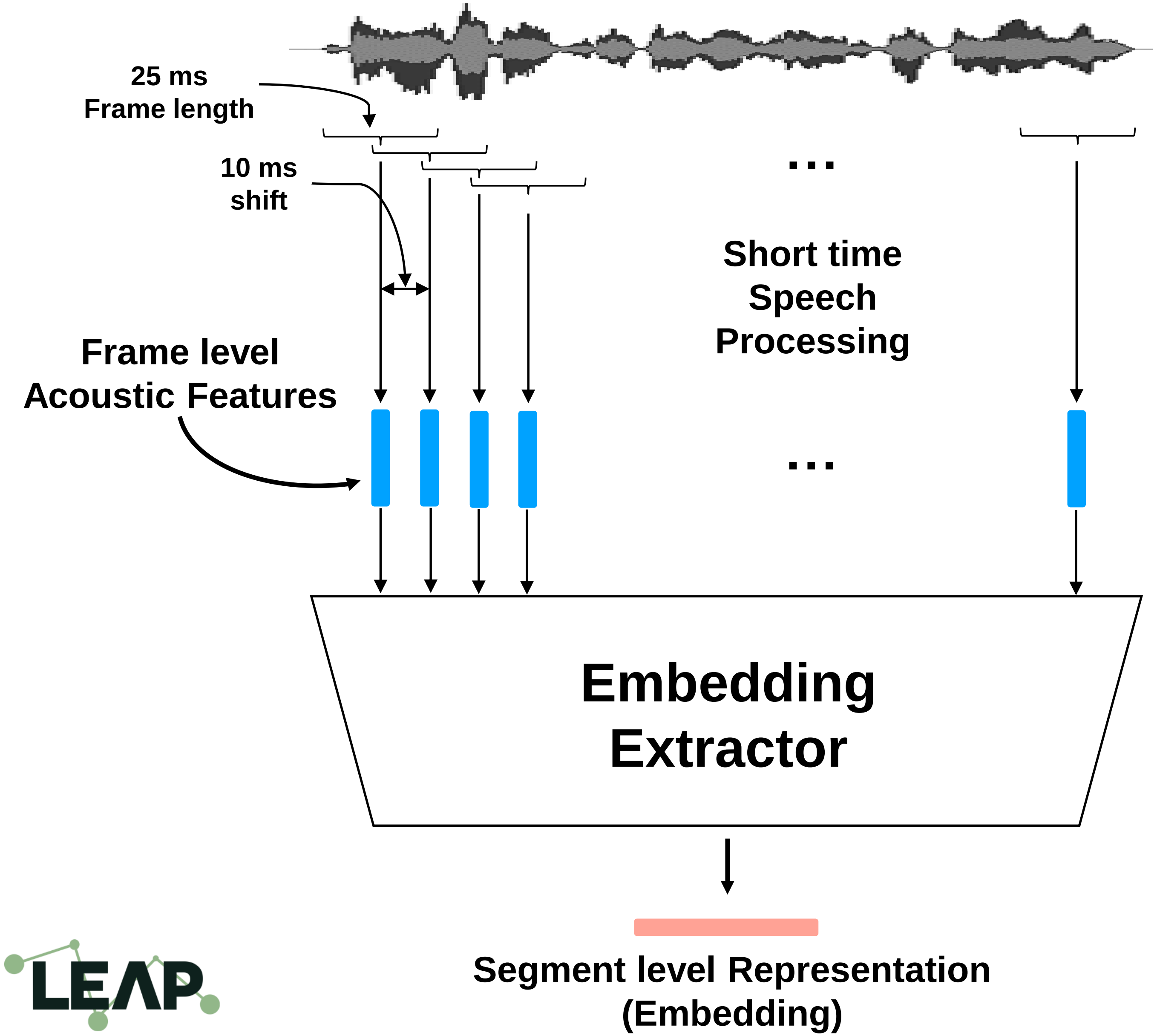
General Pipeline of NN based ASV

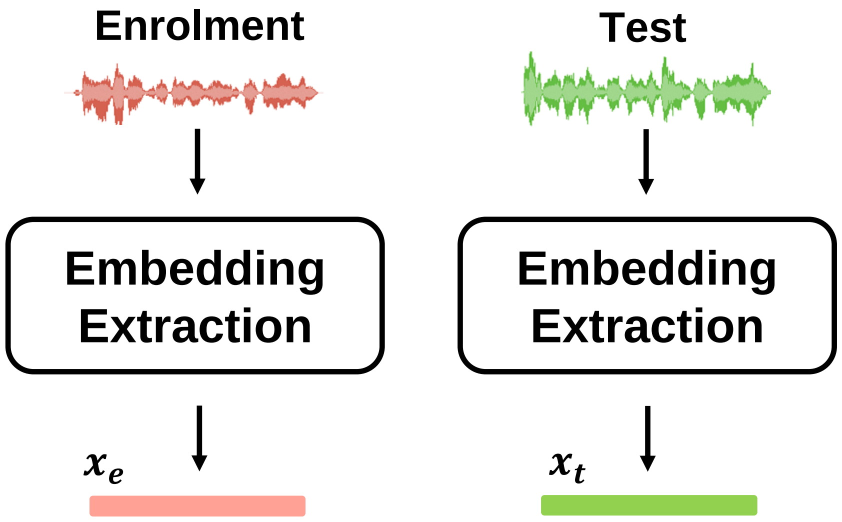


General Pipeline of NN based ASV

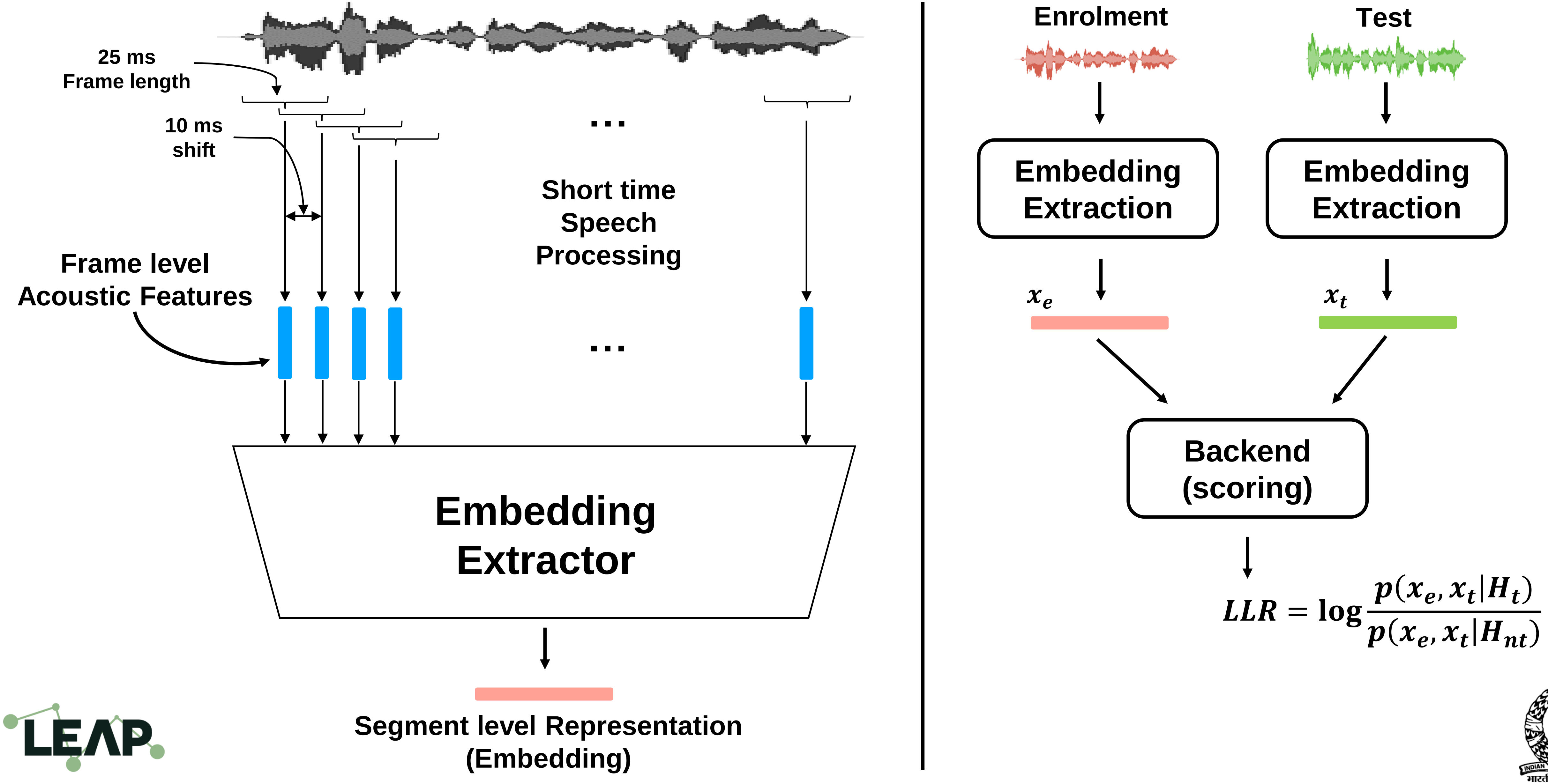


General Pipeline of NN based ASV

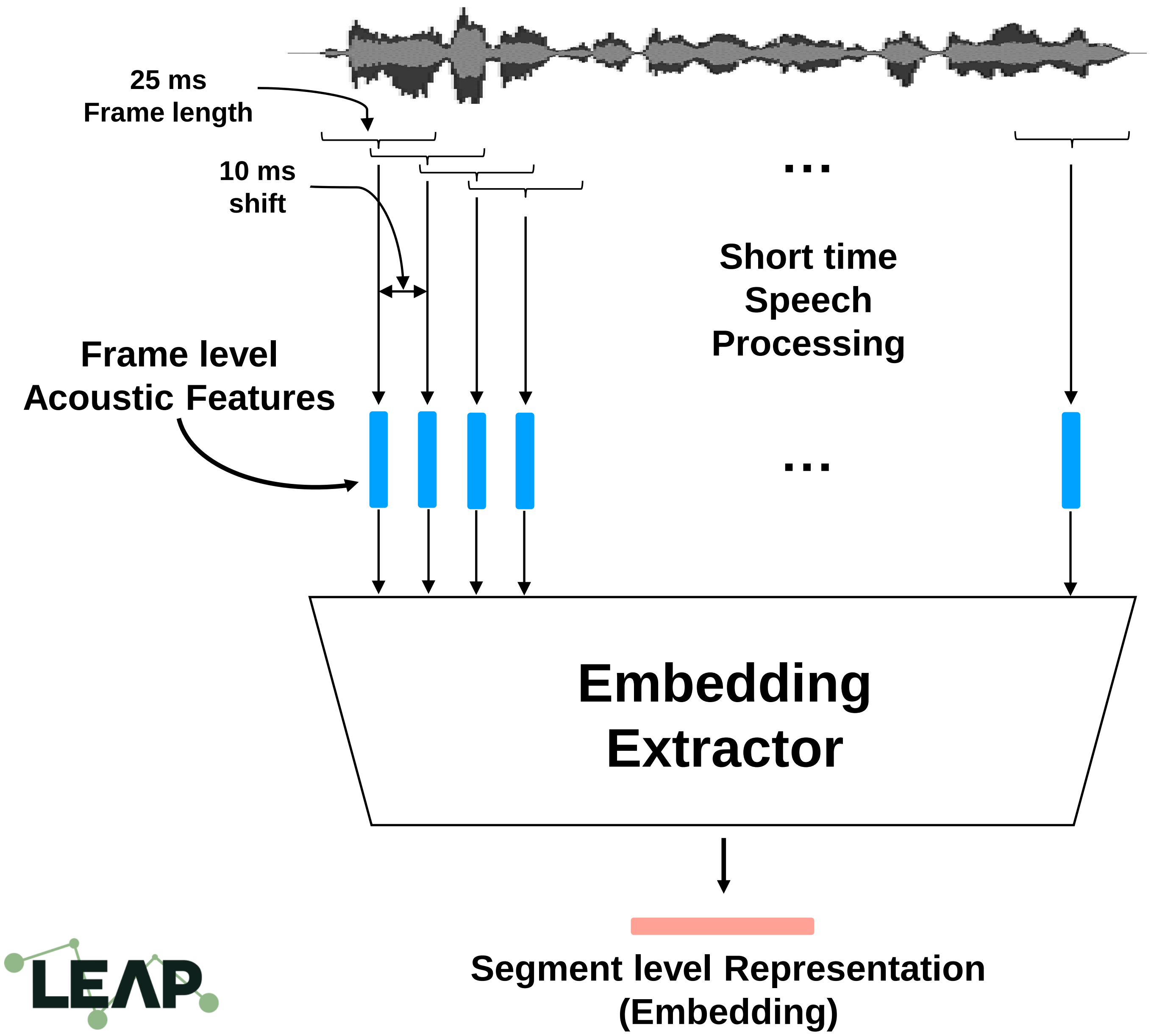


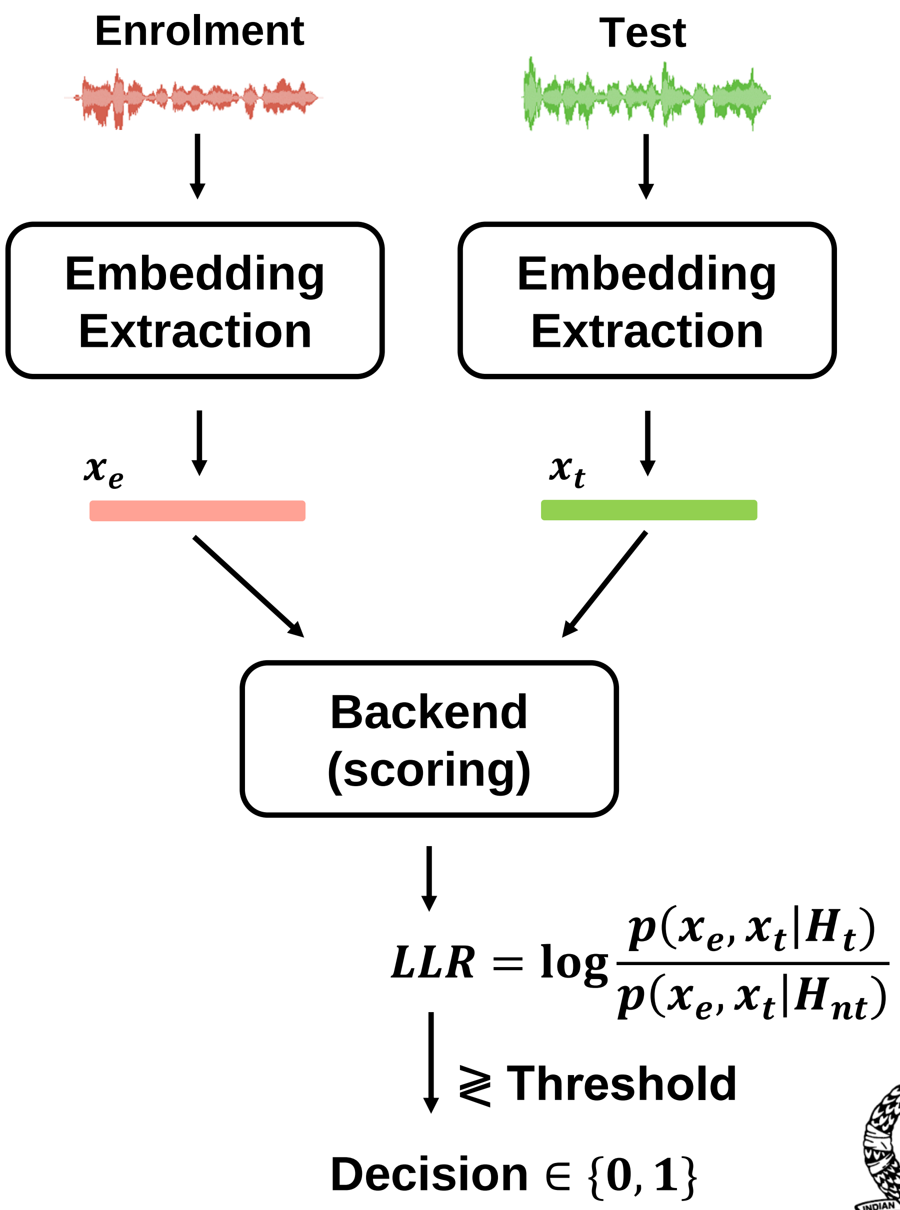


General Pipeline of NN based ASV

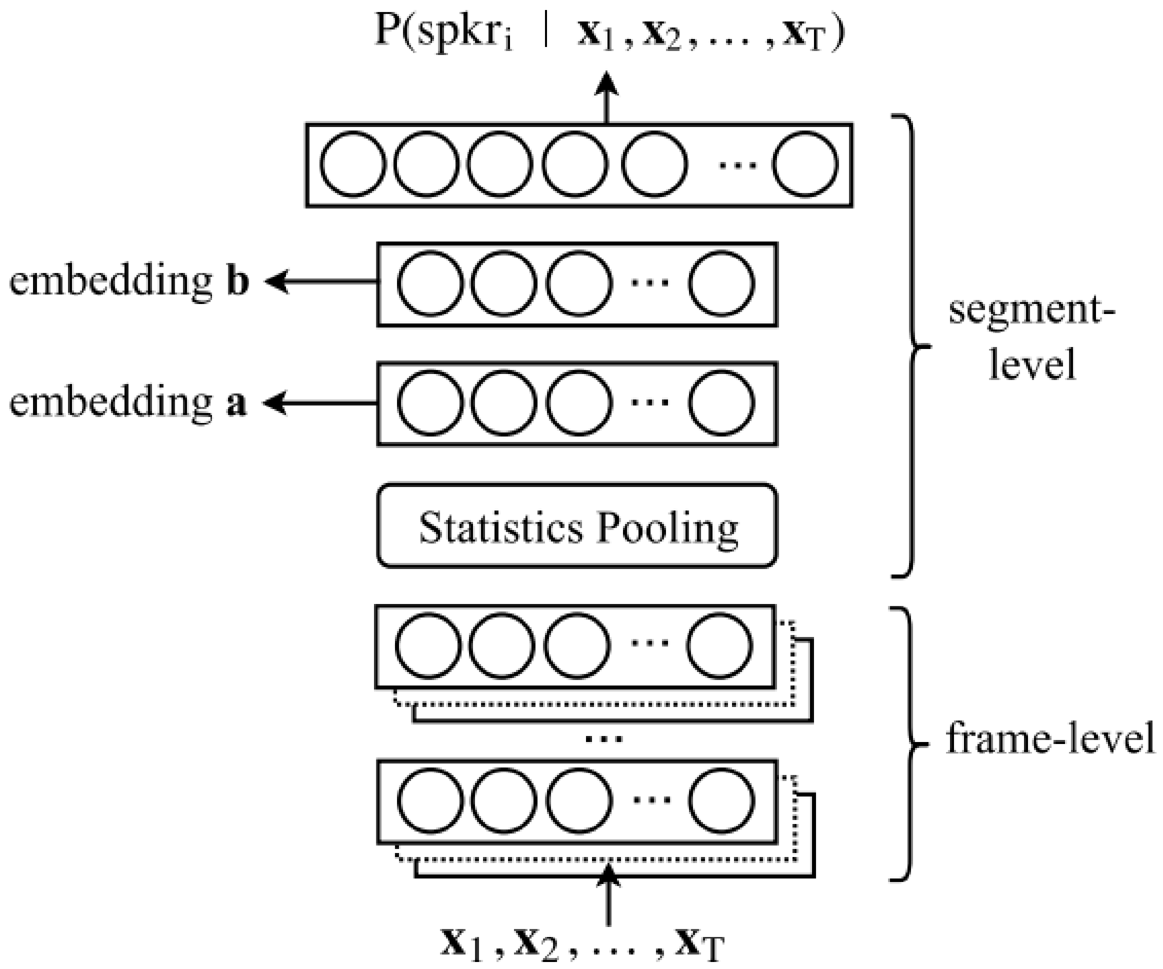


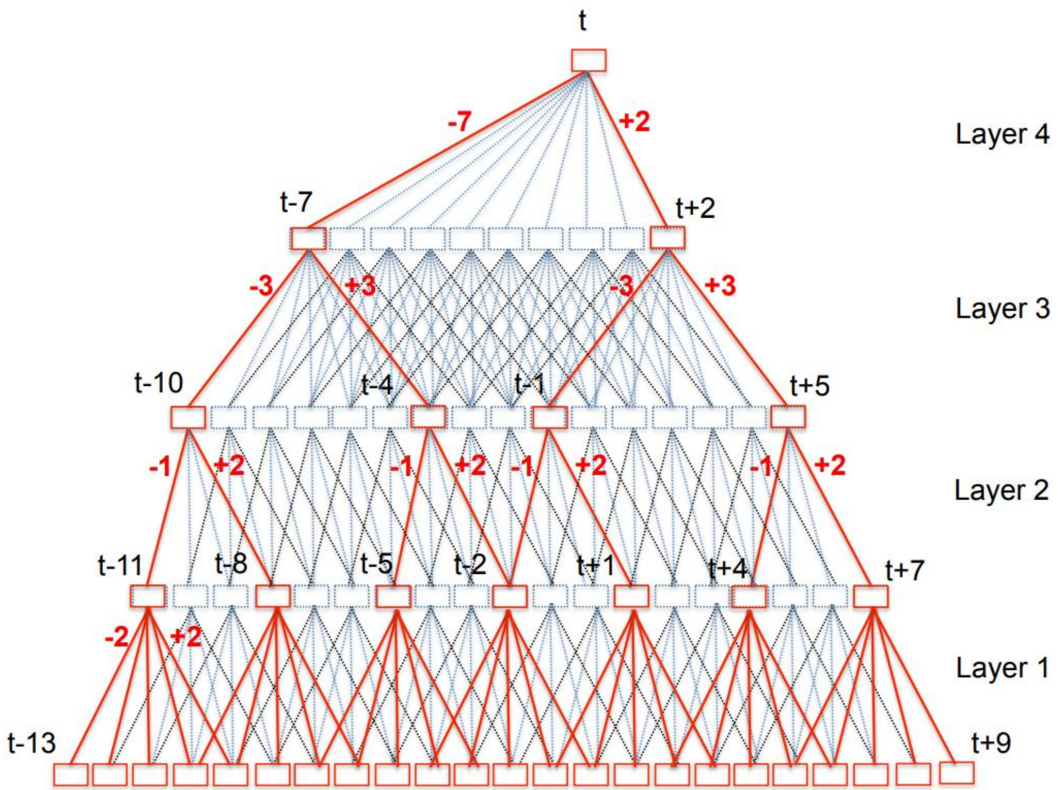
General Pipeline of NN based ASV





The X-vector Model





Snyder et.al., “X-vectors: Robust DNN Embeddings for Speaker Recognition”, 2018

Gaussian Probabilistic LDA

S. Ioffe, “*Probabilistic Linear Discriminant Analysis*”, 2006

Gaussian Probabilistic LDA

$\boldsymbol{x_r}$

S. Ioffe, “*Probabilistic Linear Discriminant Analysis*”, 2006

Gaussian Probabilistic LDA

x_r

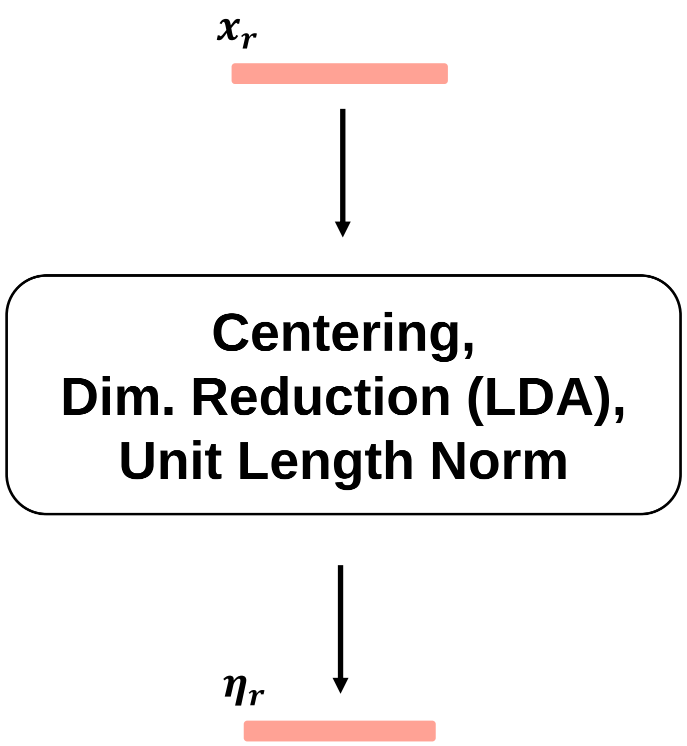
↓

Centering,
Dim. Reduction (LDA),
Unit Length Norm

↓

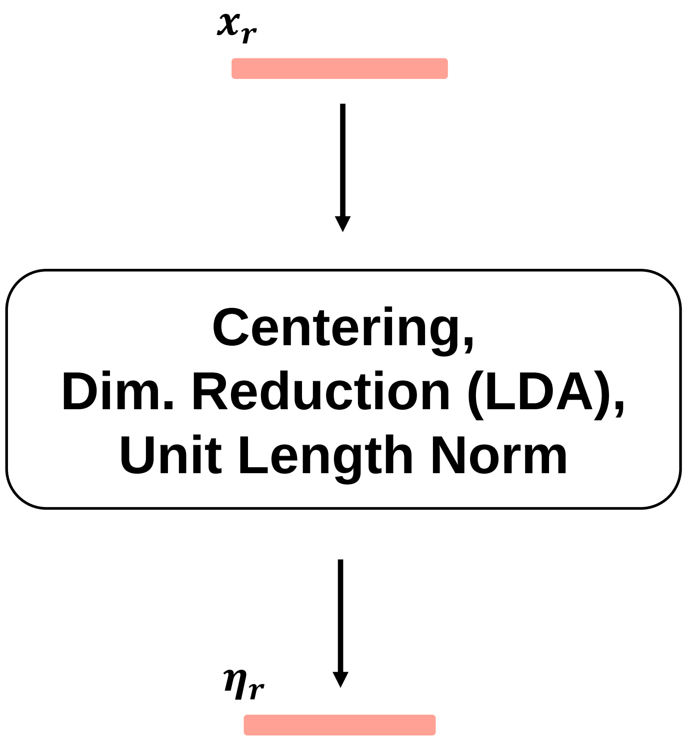
S. Ioffe, "*Probabilistic Linear Discriminant Analysis*", 2006

Gaussian Probabilistic LDA



S. Ioffe, “*Probabilistic Linear Discriminant Analysis*”, 2006

Gaussian Probabilistic LDA



$$\eta_r = \Phi\omega + \epsilon_r$$

S. Ioffe, "*Probabilistic Linear Discriminant Analysis*", 2006

Gaussian Probabilistic LDA

x_r

Centering,
Dim. Reduction (LDA),
Unit Length Norm

η_r

$$\boldsymbol{\eta_r} = \boldsymbol{\Phi\omega} + \boldsymbol{\epsilon_r}$$

Embedding

S. Ioffe, “*Probabilistic Linear Discriminant Analysis*”, 2006

Gaussian Probabilistic LDA

x_r

Centering,
Dim. Reduction (LDA),
Unit Length Norm

η_r

$$\boldsymbol{\eta_r = \Phi\omega + \epsilon_r}$$

Embedding

Latent
Speaker
Factor
$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

S. Ioffe, “*Probabilistic Linear Discriminant Analysis*”, 2006

Gaussian Probabilistic LDA

x_r

$\downarrow$

Centering,
Dim. Reduction (LDA),
Unit Length Norm

$\downarrow$

η_r

$$\boldsymbol{\eta_r} = \boldsymbol{\Phi\omega} + \boldsymbol{\epsilon_r}$$

Embedding

Speaker Subspace matrix

Latent Speaker Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

S. Ioffe, "*Probabilistic Linear Discriminant Analysis*", 2006

Gaussian Probabilistic LDA

x_r

$\downarrow$

Centering,
Dim. Reduction (LDA),
Unit Length Norm

$\downarrow$

η_r

$$\boldsymbol{\eta_r} = \boldsymbol{\Phi\omega} + \boldsymbol{\epsilon_r}$$

Embedding ($\boldsymbol{\eta_r}$)

Speaker Subspace matrix ($\boldsymbol{\Phi}$)

Latent Speaker Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ ($\boldsymbol{\omega}$)

Channel Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ ($\boldsymbol{\epsilon_r}$)

S. Ioffe, "*Probabilistic Linear Discriminant Analysis*", 2006

Gaussian Probabilistic LDA

x_r

Centering,
Dim. Reduction (LDA),
Unit Length Norm

η_r

$$\eta_r = \Phi\omega + \epsilon_r$$

Embedding

Speaker Subspace matrix

Latent Speaker Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Channel Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$

- **GPLDA Parameters: Φ and Σ Estimated using E-M Algorithm**

S. Ioffe, "*Probabilistic Linear Discriminant Analysis*", 2006

Gaussian Probabilistic LDA

x_r

Centering,
Dim. Reduction (LDA),
Unit Length Norm

η_r

$$\boldsymbol{\eta_r} = \boldsymbol{\Phi\omega} + \boldsymbol{\epsilon_r}$$

Embedding

Speaker Subspace matrix

Latent Speaker Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Channel Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$

- **GPLDA Parameters: $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$ Estimated using E-M Algorithm**

LLR:

$$\boxed{l(\eta_e, \eta_t) = \eta_e^T Q \eta_e + \eta_t^T Q \eta_t + 2\, \eta_e^T P \eta_t + const}$$

S. Ioffe, "*Probabilistic Linear Discriminant Analysis*", 2006

Gaussian Probabilistic LDA

x_r

Centering,
Dim. Reduction (LDA),
Unit Length Norm

η_r

$$\eta_r = \Phi\omega + \epsilon_r$$

Embedding

Speaker Subspace matrix

Latent Speaker Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Channel Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$

- **GPLDA Parameters: Φ and Σ Estimated using E-M Algorithm**

LLR:

$$\boxed{l(\eta_e, \eta_t) = \eta_e^T Q \eta_e + \eta_t^T Q \eta_t + 2\,\eta_e^T P \eta_t + const}$$

where $Q = \Sigma_{tot}^{-1} - \left(\Sigma_{tot} - \Sigma_{ac}\Sigma_{tot}^{-1}\Sigma_{ac}\right)^{-1}$

$$P = \Sigma_{tot}^{-1}\,\Sigma_{ac}\left(\Sigma_{tot} - \Sigma_{ac}\Sigma_{tot}^{-1}\Sigma_{ac}\right)^{-1}$$

$\Sigma_{tot} = \Phi\Phi^T + \Sigma$ **and** $\Sigma_{tot} = \Phi\Phi^T$

S. Ioffe, "*Probabilistic Linear Discriminant Analysis*", 2006

Gaussian Probabilistic LDA

x_r

Centering,
Dim. Reduction (LDA),
Unit Length Norm

η_r

$$\boldsymbol{\eta_r = \Phi\omega + \epsilon_r}$$

Embedding

Speaker Subspace matrix

Latent Speaker Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Channel Factor $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$

- **GPLDA Parameters: $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$ Estimated using E-M Algorithm**

LLR:

$$\boxed{l(\eta_e, \eta_t) = \eta_e^T Q \eta_e + \eta_t^T Q \eta_t + 2\, \eta_e^T P \eta_t + const}$$

Some quadratic function of η_e and η_t

where $Q = \Sigma_{tot}^{-1} - \left(\Sigma_{tot} - \Sigma_{ac}\Sigma_{tot}^{-1}\Sigma_{ac}\right)^{-1}$

$P = \Sigma_{tot}^{-1}\, \Sigma_{ac} \left(\Sigma_{tot} - \Sigma_{ac}\Sigma_{tot}^{-1}\Sigma_{ac}\right)^{-1}$

$\Sigma_{tot} = \Phi\Phi^T + \Sigma$ **and** $\Sigma_{tot} = \Phi\Phi^T$

S. Ioffe, “*Probabilistic Linear Discriminant Analysis*”, 2006

Neural PLDA (NPLDA)

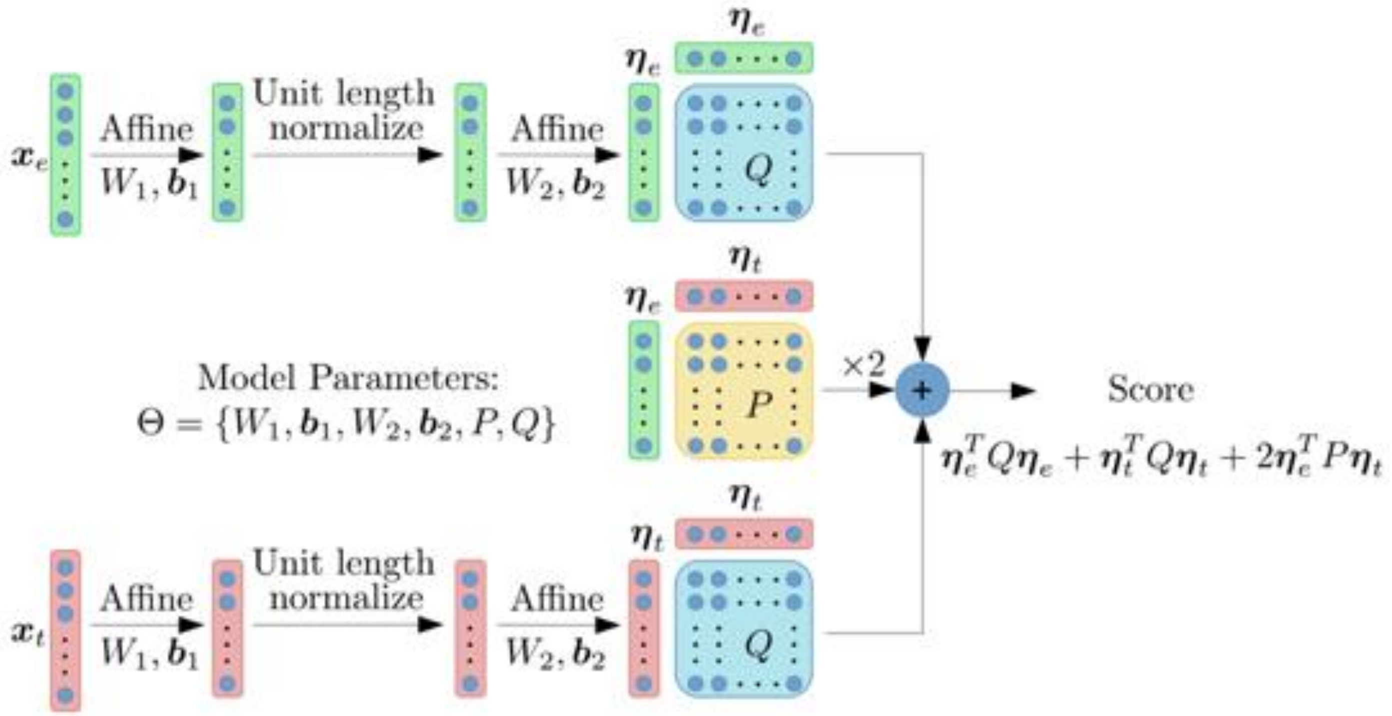
Ramoji et al., “*NPLDA: A Deep Neural PLDA model for speaker verification.*”, Odyssey 2020

Neural PLDA (NPLDA)

- NPLDA is a **pairwise discriminative network**, where we construct the pre-processing steps of LDA as an affine layer, unit-length normalization as a non-linear activation.

Neural PLDA (NPLDA)

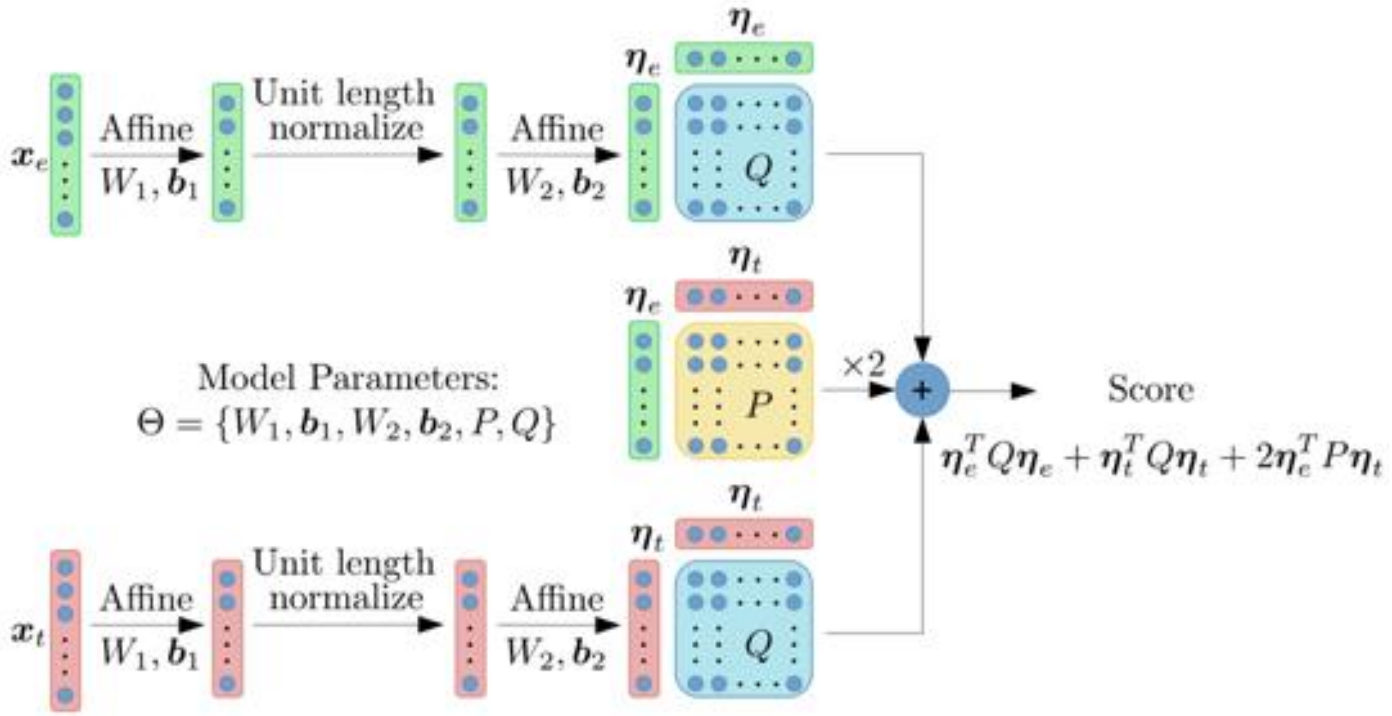
- NPLDA is a **pairwise discriminative network**, where we construct the pre-processing steps of LDA as an affine layer, unit-length normalization as a non-linear activation.



Ramoji et al., “*NPLDA: A Deep Neural PLDA model for speaker verification.*”, Odyssey 2020

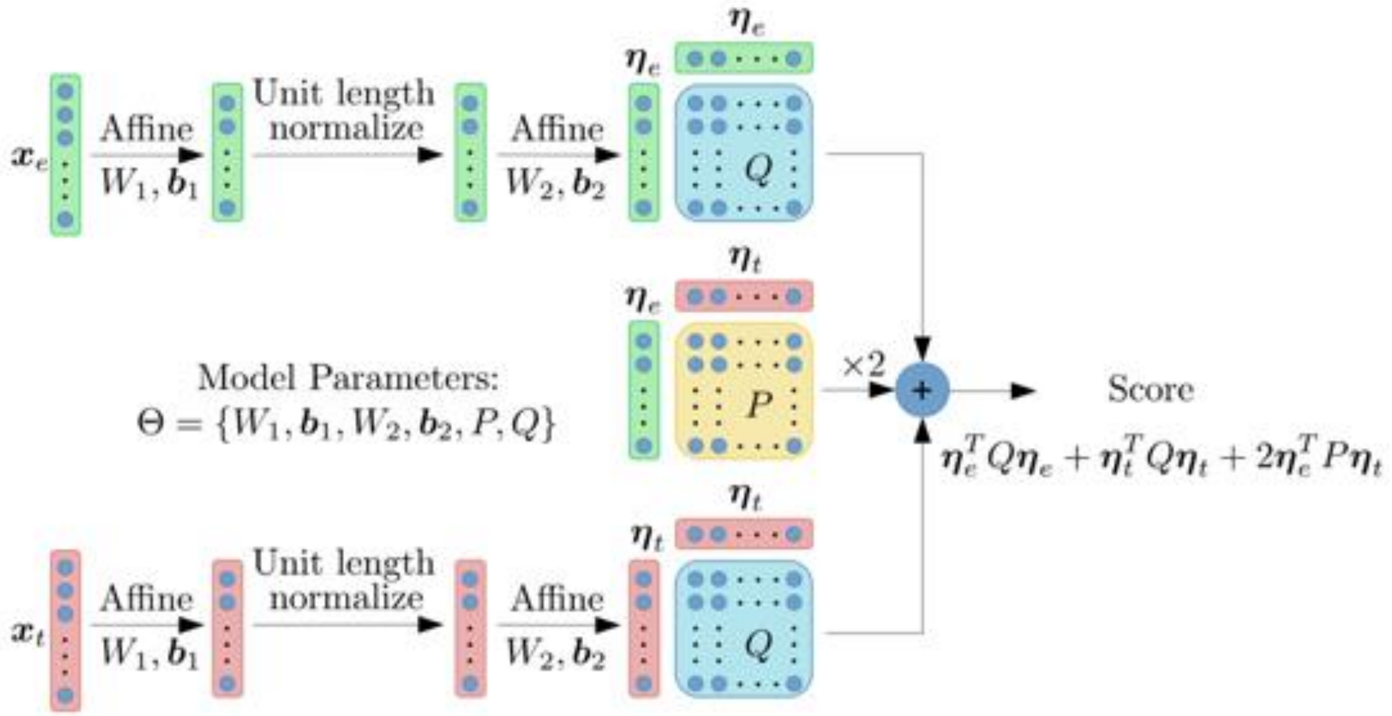
Neural PLDA (NPLDA)

- NPLDA is a **pairwise discriminative network**, where we construct the pre-processing steps of LDA as an affine layer, unit-length normalization as a non-linear activation.
- The final PLDA pair-wise scoring implemented as a Quadratic layer.



Neural PLDA (NPLDA)

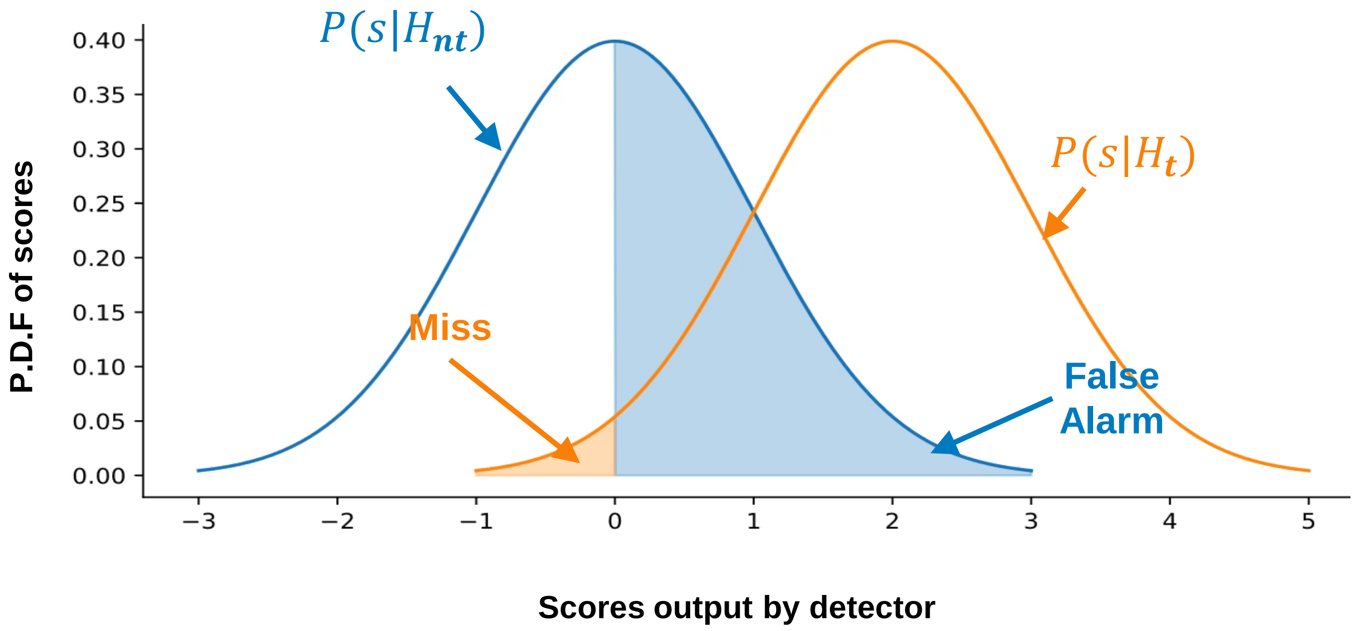
- NPLDA is a **pairwise discriminative network**, where we construct the pre-processing steps of LDA as an affine layer, unit-length normalization as a non-linear activation.
- The final PLDA pair-wise scoring implemented as a Quadratic layer.
- Parameters are initialized with the GPLDA model weights.



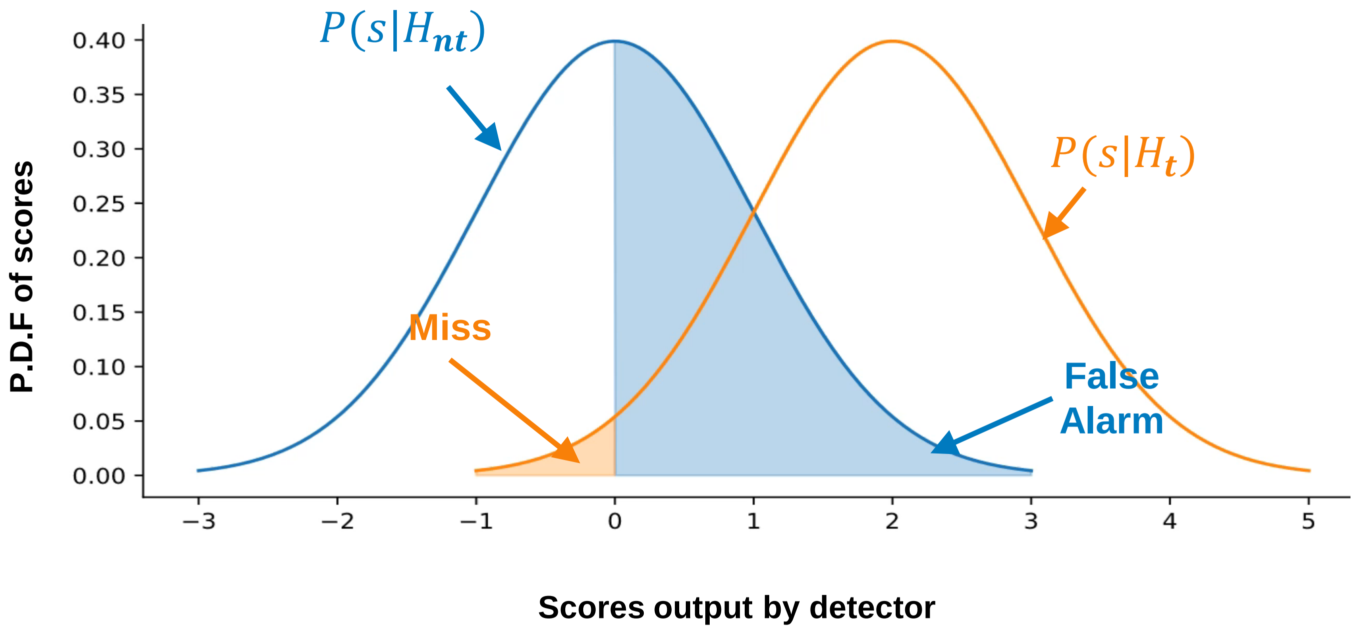
Ramoji et al., “*NPLDA: A Deep Neural PLDA model for speaker verification.*”, Odyssey 2020

Evaluation Metrics

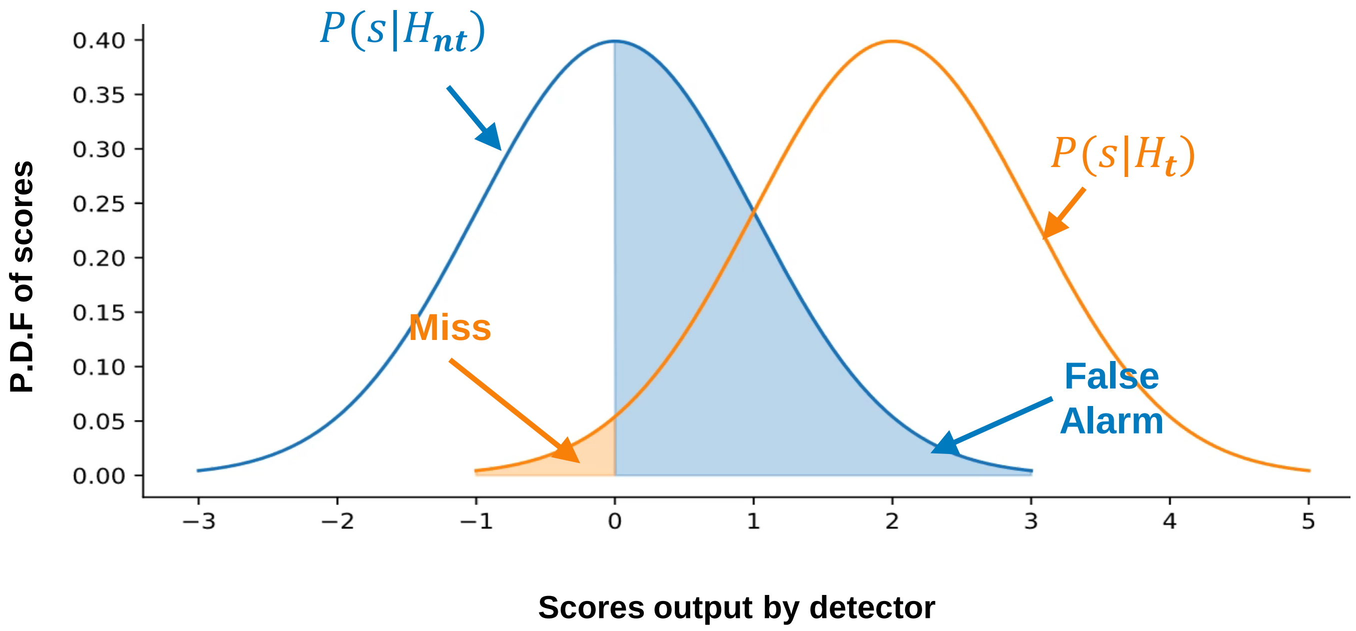
Evaluation Metrics



Evaluation Metrics

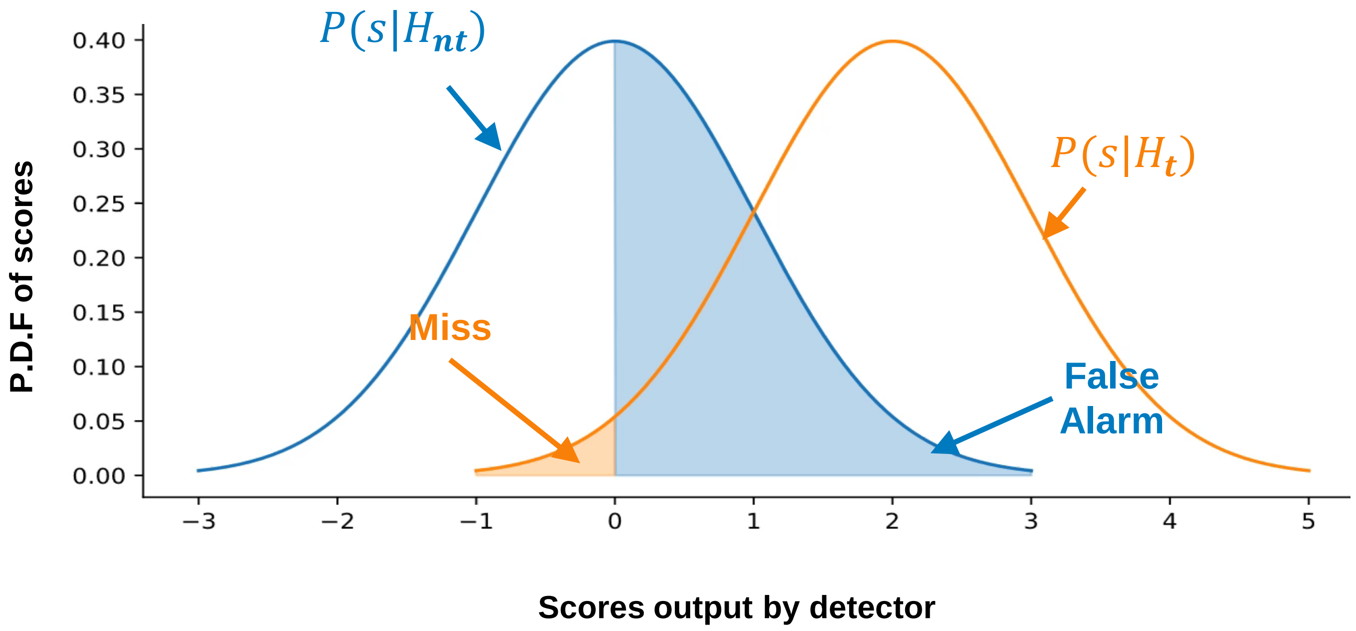


Evaluation Metrics



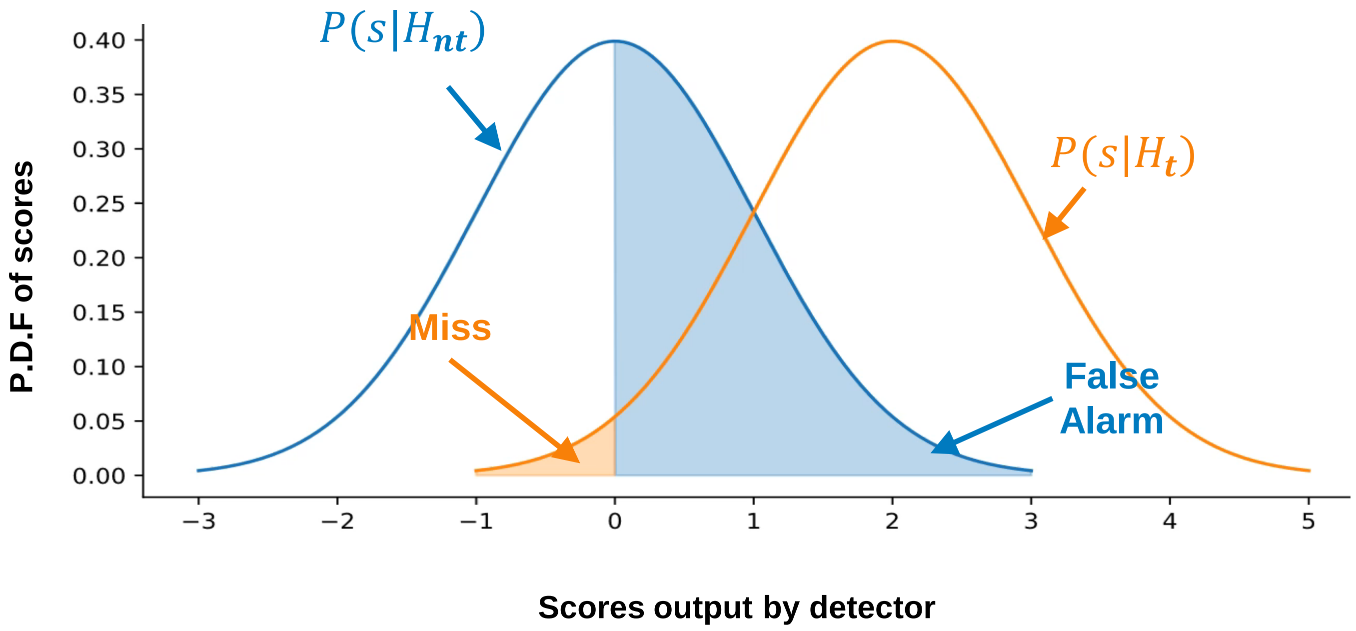
$$\text{DCF (Bayes risk)} = C_{Miss} \times P_{Target} \times P_{Miss}(\theta) + C_{FA} \times (1 - P_{Target}) \times P_{FA}(\theta)$$

Evaluation Metrics



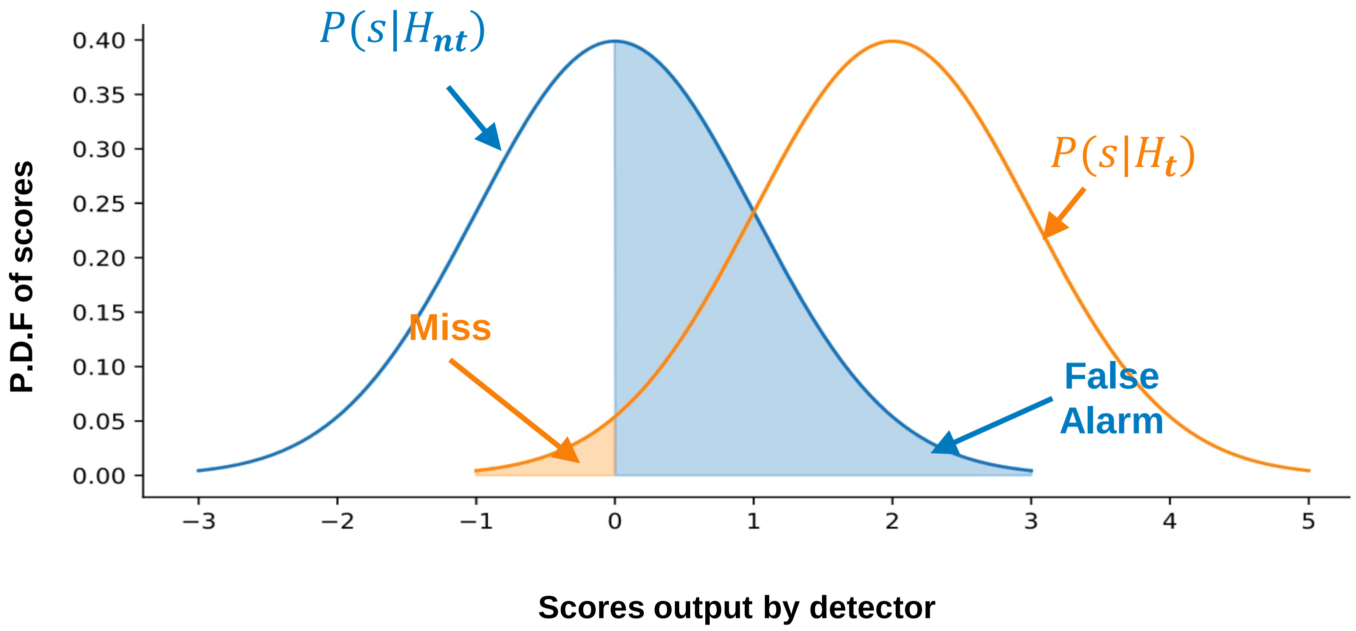
$$Normalized\ DCF = \frac{C_{Miss} \times P_{Target}}{C_{Miss} \times P_{Target}} \times P_{Miss}(\theta) + \frac{C_{FA} \times (1 - P_{Target})}{C_{Miss} \times P_{Target}} \times P_{FA}(\theta)$$

Evaluation Metrics



$$Normalized\ DCF = \underbrace{\frac{C_{Miss} \times P_{Target}}{C_{Miss} \times P_{Target}}}_{=1} \times P_{Miss}(\theta) + \underbrace{\frac{C_{FA} \times (1 - P_{Target})}{C_{Miss} \times P_{Target}}}_{=\beta} \times P_{FA}(\theta)$$

Evaluation Metrics

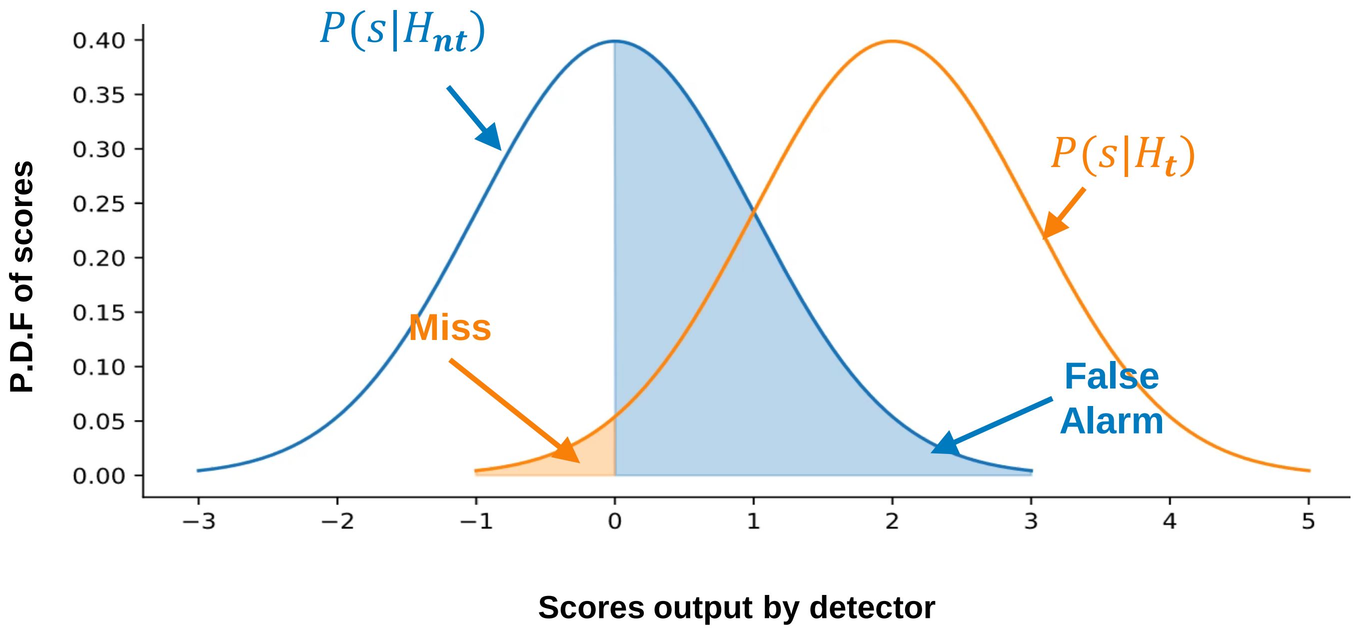


- Normalized DCF: $C_{Norm}(\beta,\theta) = P_{Miss}(\theta) + \beta P_{FA}(\theta)$

  where $\beta = \frac{C_{FA}(1-P_{Target})}{C_{Miss}\,P_{Target}}$; $P_{Miss}(\theta) = \frac{1}{\mathcal{T}}\sum_{i\in\mathcal{T}} \mathbb{1}(s_i < \theta)$; $P_{FA}(\theta) = \frac{1}{\mathcal{N}}\sum_{i\in N} \mathbb{1}(s_i \geq \theta)$

- minDCF: $C_{Min}(\beta) = \underset{\theta}{\operatorname{argmin}}\, P_{Miss}(\theta) + \beta P_{FA}(\theta)$

Evaluation Metrics



- Normalized DCF: $C_{Norm}(\beta, \theta) = P_{Miss}(\theta) + \beta P_{FA}(\theta)$

  where $\beta = \frac{C_{FA}(1-P_{Target})}{C_{Miss}\,P_{Target}}$; $P_{Miss}(\theta) = \frac{1}{\mathcal{T}} \sum_{i \in \mathcal{T}} \mathbb{I}(s_i < \theta)$; $P_{FA}(\theta) = \frac{1}{\mathcal{N}} \sum_{i \in N} \mathbb{I}(s_i \geq \theta)$

- minDCF: $C_{Min}(\beta) = \underset{\theta}{\operatorname{argmin}}\, P_{Miss}(\theta) + \beta P_{FA}(\theta)$

- Equal Error Rate (EER)

Loss Functions

Loss Functions

- Soft Detection Cost Function (SoftDCF): Loss function for NPLDA Model training

$$C_{Norm}^{(soft)}(\beta, \theta) = P_{Miss}^{(soft)}(\theta) + \beta P_{FA}^{(soft)}(\theta)$$

Loss Functions

- Soft Detection Cost Function (SoftDCF): Loss function for NPLDA Model training

$$C_{Norm}^{(soft)}(\beta,\theta) = P_{Miss}^{(soft)}(\theta) + \beta P_{FA}^{(soft)}(\theta)$$

$$P_{Miss}^{(soft)}(\theta) = \frac{1}{\mathcal{T}}\sum_{i\in\mathcal{T}}[1-\sigma(\alpha(s_i-\theta))] \quad ; \qquad P_{FA}^{(soft)}(\theta) = \frac{1}{\mathcal{N}}\sum_{i\in N}\sigma(\alpha(s_i-\theta))$$

Loss Functions

- Soft Detection Cost Function (SoftDCF): Loss function for NPLDA Model training

$$C_{Norm}^{(soft)}(\beta,\theta) = P_{Miss}^{(soft)}(\theta) + \beta P_{FA}^{(soft)}(\theta)$$

$$P_{Miss}^{(soft)}(\theta) = \frac{1}{\mathcal{T}}\sum_{i\in\mathcal{T}}[1-\sigma(\alpha(s_i-\theta))] \quad ; \qquad P_{FA}^{(soft)}(\theta) = \frac{1}{\mathcal{N}}\sum_{i\in N}\sigma(\alpha(s_i-\theta))$$

- Binary Cross Entropy and its weighted versions.

$$L_{BCE} = \lambda_1 \sum_{i\in\mathcal{T}} \log\left(1+e^{-(s_i-\theta)}\right) + \lambda_2 \sum_{i\in\mathcal{N}} \log\left(1+e^{(s_i-\theta)}\right)$$

Loss Functions

- Soft Detection Cost Function (SoftDCF): Loss function for NPLDA Model training

$$C_{Norm}^{(soft)}(\beta, \theta) = P_{Miss}^{(soft)}(\theta) + \beta P_{FA}^{(soft)}(\theta)$$

$$P_{Miss}^{(soft)}(\theta) = \frac{1}{\mathcal{T}} \sum_{i \in \mathcal{T}} [1 - \sigma(\alpha(s_i - \theta))] \quad ; \qquad P_{FA}^{(soft)}(\theta) = \frac{1}{\mathcal{N}} \sum_{i \in N} \sigma(\alpha(s_i - \theta))$$

- Binary Cross Entropy and its weighted versions.

$$L_{BCE} = \lambda_1 \sum_{i \in \mathcal{T}} \log\left(1 + e^{-(s_i - \theta)}\right) \; + \; \lambda_2 \sum_{i \in \mathcal{N}} \log\left(1 + e^{(s_i - \theta)}\right)$$

  - Vanilla BCE : $\lambda_1 = \lambda_2 = 1 \quad ; \quad \theta = 0$

Loss Functions

- Soft Detection Cost Function (SoftDCF): Loss function for NPLDA Model training

$$C_{Norm}^{(soft)}(\beta,\theta) = P_{Miss}^{(soft)}(\theta) + \beta P_{FA}^{(soft)}(\theta)$$

$$P_{Miss}^{(soft)}(\theta) = \frac{1}{\mathcal{T}}\sum_{i\in\mathcal{T}}[1-\sigma(\alpha(s_i-\theta))] \quad ; \qquad P_{FA}^{(soft)}(\theta) = \frac{1}{\mathcal{N}}\sum_{i\in N}\sigma(\alpha(s_i-\theta))$$

- Binary Cross Entropy and its weighted versions.

$$L_{BCE} = \lambda_1 \sum_{i\in\mathcal{T}} \log\left(1+e^{-(s_i-\theta)}\right) + \lambda_2 \sum_{i\in\mathcal{N}} \log\left(1+e^{(s_i-\theta)}\right)$$

  - Vanilla BCE : $\lambda_1 = \lambda_2 = 1 \quad ; \quad \theta = 0$
  - Likelihood ratio cost (C_{llr}) [1] : $\lambda_1 = \frac{1}{|\mathcal{T}|} \quad ; \quad \lambda_2 = \frac{1}{|\mathcal{N}|}; \quad \theta = 0$

Loss Functions

- Soft Detection Cost Function (SoftDCF): Loss function for NPLDA Model training

$$C_{Norm}^{(soft)}(\beta,\theta) = P_{Miss}^{(soft)}(\theta) + \beta P_{FA}^{(soft)}(\theta)$$

$$P_{Miss}^{(soft)}(\theta) = \frac{1}{\mathcal{T}}\sum_{i\in\mathcal{T}}[1-\sigma(\alpha(s_i-\theta))] \quad ; \qquad P_{FA}^{(soft)}(\theta) = \frac{1}{\mathcal{N}}\sum_{i\in N}\sigma(\alpha(s_i-\theta))$$

- Binary Cross Entropy and its weighted versions.

$$L_{BCE} = \lambda_1 \sum_{i\in\mathcal{T}} \log\left(1+e^{-(s_i-\theta)}\right) + \lambda_2 \sum_{i\in\mathcal{N}} \log\left(1+e^{(s_i-\theta)}\right)$$

  - Vanilla BCE : $\lambda_1 = \lambda_2 = 1 \quad ; \quad \theta = 0$
  - Likelihood ratio cost (C_{llr}) [1] : $\lambda_1 = \frac{1}{|\mathcal{T}|} \quad ; \quad \lambda_2 = \frac{1}{|\mathcal{N}|}; \quad \theta = 0$
  - Weighted Cllr [2] : $\lambda_1 = \frac{C_{Miss}P_{Target}}{|\mathcal{T}|} \quad ; \quad \lambda_2 = \frac{C_{FA}(1-P_{Target})}{|\mathcal{N}|}; \quad \theta = \log\beta$

[1] N Brümmer et. al., “Application Independent Evaluation of speaker detection
[2] L. Ferrer et. al., “*A discriminative condition-aware backend for speaker verification*” ICASSP 2020

Experiments

- **Train dataset Description: *VoxCeleb (Parts 1 and 2)***
  - Large, Publicly available dataset containing speech extracted from celebrity interview videos available on YouTube.

Experiments

- **Train dataset Description: *VoxCeleb (Parts 1 and 2)***
  - Large, Publicly available dataset containing speech extracted from celebrity interview videos available on YouTube.
  - Spans a wide range of different ethnicity, accents, professions, and ages.

Experiments

- **Train dataset Description: *VoxCeleb (Parts 1 and 2)***
  - Large, Publicly available dataset containing speech extracted from celebrity interview videos available on YouTube.
  - Spans a wide range of different ethnicity, accents, professions, and ages.
  - Contains around **1.2M utterances from 7323 speakers**.
  - An extended TDNN (E-TDNN) model was trained with the softmax-cross-entropy speaker classification objective, from which 512 dim speaker embeddings were extracted.

Experiments

- **Train dataset Description: *VoxCeleb (Parts 1 and 2)***
  - Large, Publicly available dataset containing speech extracted from celebrity interview videos available on YouTube.
  - Spans a wide range of different ethnicity, accents, professions, and ages.
  - Contains around **1.2M utterances from 7323 speakers**.
  - An extended TDNN (E-TDNN) model was trained with the softmax-cross-entropy speaker classification objective, from which 512 dim speaker embeddings were extracted.
  - A 5-fold augmentation strategy is used that adds four corrupted copies of the original recordings to the training – Clean, Noise, Babble, Music, Reverberation

Experiments on NPLDA

- **Test Datasets Description:**
  - ***SITW (Speakers in the Wild) Dataset:***
    - Consists of ***300 speakers*** across clean interviews, red carpet interviews, stadium conditions, indoor and outdoor conditions.
    - The evaluation set consists of ***721,788 trials***
  - ***VOiCES Challenge 2019 Dataset:***
    - Re-recorded read speech in acoustically challenging, noisy and reverberent environments.
    - The Development set contains 16k segments 196 speakers, and 4 Million trials
    - The Test set contains 11k segments from 100 speakers, 3.6 Million trials

Results on NPLDA

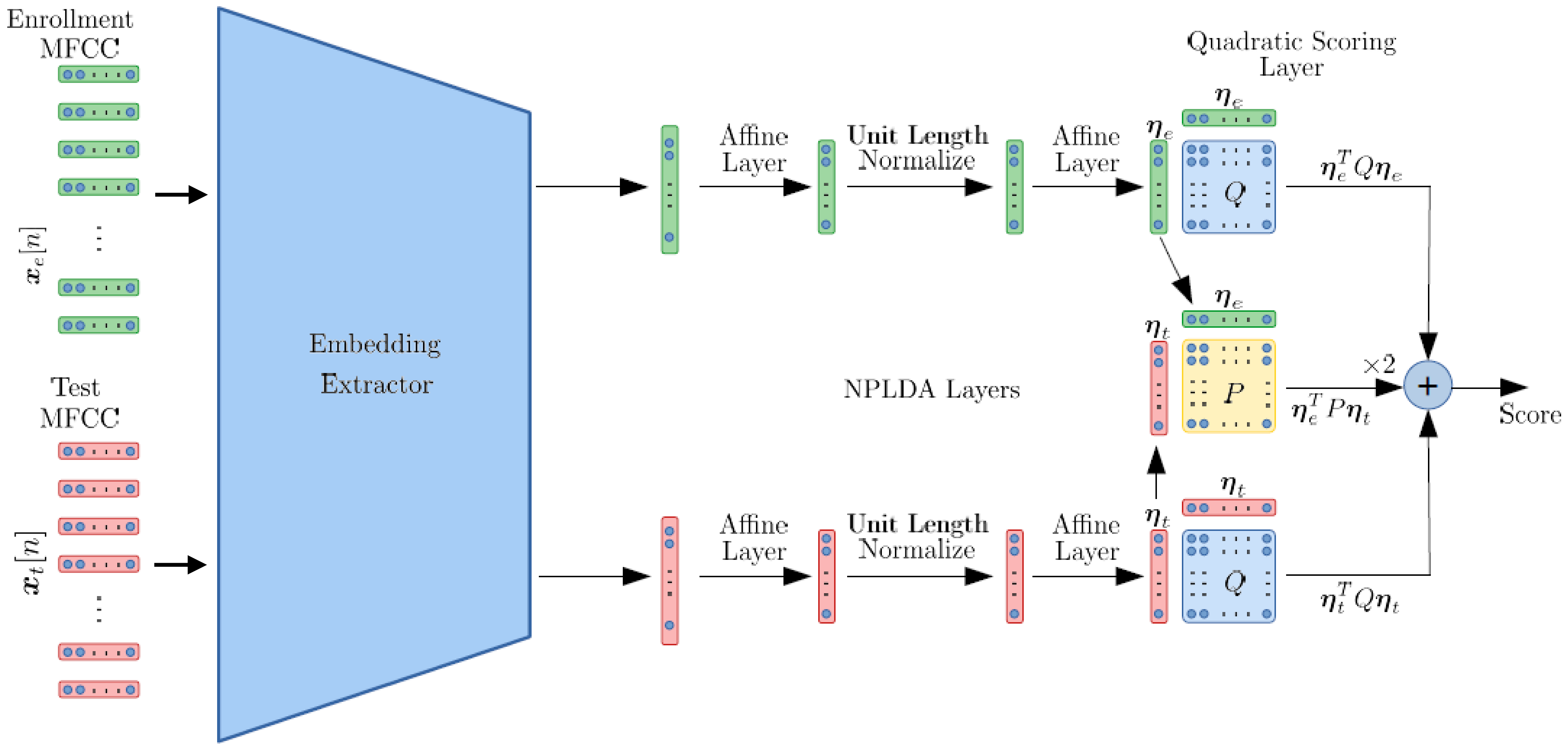
Model	PLDA Train Dataset	SITW Core-Core		VOiCES Dev		VOiCES Eval	
		EER (%)	minDCF	EER (%)	minDCF	EER (%)	minDCF
GPLDA	VoxCeleb	2.79	0.29	2.79	0.31	7.35	0.57
GPLDA	VoxCeleb Augmented	2.79	0.29	2.79	0.30	6.38	0.53
Gaussian Backend	VoxCeleb	3.19	0.31	3.14	0.33	7.58	0.60
Gaussian Backend	VoxCeleb Augmented	3.06	0.31	2.89	0.30	6.63	0.53
DPLDA	VoxCeleb Augmented	2.98	0.32	3.05	0.36	6.65	0.56
NPLDA	VoxCeleb	2.14	0.23	2.20	0.26	6.72	0.51
NPLDA	VoxCeleb Augmented	**2.05**	**0.20**	**1.91**	**0.23**	**6.01**	**0.49**
NPLDA (BCE Loss)	VoxCeleb Augmented	2.10	0.22	2.32	0.26	6.34	0.53

Table 1: Performance of systems on SITW Eval Core-Core, VOiCES Dev and VOiCES Eval using the GPLDA baseline model, Gaussian backend, Discriminative PLDA (DPLDA) and the proposed NPLDA model. We also report the use of binary cross-entropy (BCE) Loss in the NPLDA model place of the soft detection cost. The best scores are highlighted.

- Relative improvements in the range of 9% to 23% in terms of EER, and 11% to 31%in terms of minDCF.

Ramoji et al., “*NPLDA: A Deep Neural PLDA model for speaker verification.*”, Odyssey 2020

End-to-End Siamese NPLDA Model



Implementation Challenges

Implementation Challenges

- The NPLDA Back-end model was trained on the segment level embeddings (x-vectors) which are pre-extracted and stored on the disk.

Implementation Challenges

- The NPLDA Back-end model was trained on the segment level embeddings (x-vectors) which are pre-extracted and stored on the disk.

- The NPLDA back-end with the softDCF loss function was found to achieve the best results with a batch size of 2048 trials (4096 unique x-vectors).

Implementation Challenges

- The NPLDA Back-end model was trained on the segment level embeddings (x-vectors) which are pre-extracted and stored on the disk.
- The NPLDA back-end with the softDCF loss function was found to achieve the best results with a batch size of 2048 trials (4096 unique x-vectors).
- The Siamese E2E-NPLDA model requires frame-level processing of the MFCC features at 100 frames per second of the audio (as opposed to the segment level x-vectors).
    - We need to pad/truncate the audio to a fixed duration, in every batch.

Implementation Challenges

- The NPLDA Back-end model was trained on the segment level embeddings (x-vectors) which are pre-extracted and stored on the disk.
- The NPLDA back-end with the softDCF loss function was found to achieve the best results with a batch size of 2048 trials (4096 unique x-vectors).
- The Siamese E2E-NPLDA model requires frame-level processing of the MFCC features at 100 frames per second of the audio (as opposed to the segment level x-vectors).
    - We need to pad/truncate the audio to a fixed duration, in every batch.
- The GPU requirements for the NPLDA back-end was <600 Mb

Implementation Challenges

- The NPLDA Back-end model was trained on the segment level embeddings (x-vectors) which are pre-extracted and stored on the disk.

- The NPLDA back-end with the softDCF loss function was found to achieve the best results with a batch size of 2048 trials (4096 unique x-vectors).

- The Siamese E2E-NPLDA model requires frame-level processing of the MFCC features at 100 frames per second of the audio (as opposed to the segment level x-vectors).
    - We need to pad/truncate the audio to a fixed duration, in every batch.

- The GPU requirements for the NPLDA back-end was <600 Mb

- The GPU requirements for the E2E-NPLDA increases linearly with total audio duration per batch

    - Was estimated to be >200GB for 2048 trials of 20 sec segments.

Implementation Challenges

Implementation Challenges

- As we had a GPU limit of 16 Gb, the neural network architecture we used could process a total audio duration of 130 secs per batch (~64 utterances of 20 secs each)

Implementation Challenges

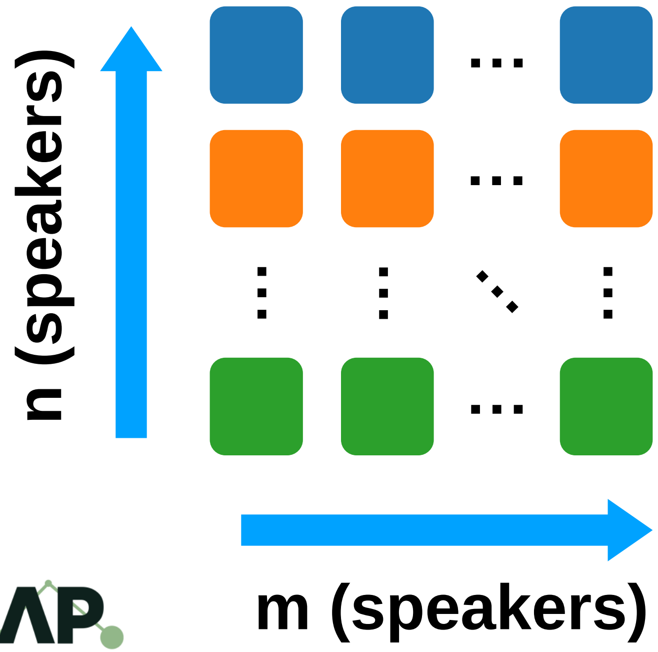
- As we had a GPU limit of 16 Gb, the neural network architecture we used could process a total audio duration of 130 secs per batch (~64 utterances of 20 secs each)

- To confine the E2E-NPLDA to meet the GPU memory constraints, but also have >2k trials per batch, we used a low-memory trial sampling algorithm:

Implementation Challenges

- As we had a GPU limit of 16 Gb, the neural network architecture we used could process a total audio duration of 130 secs per batch (~64 utterances of 20 secs each)

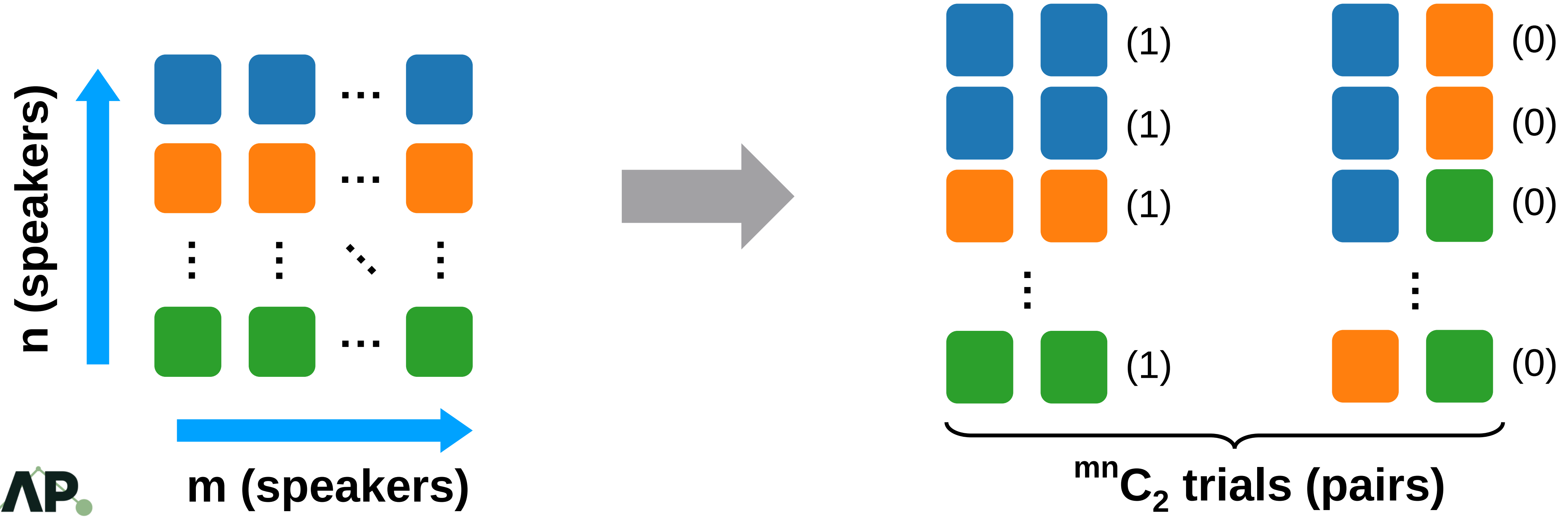
- To confine the E2E-NPLDA to meet the GPU memory constraints, but also have >2k trials per batch, we used a low-memory trial sampling algorithm:



Implementation Challenges

- As we had a GPU limit of 16 Gb, the neural network architecture we used could process a total audio duration of 130 secs per batch (~64 utterances of 20 secs each)

- To confine the E2E-NPLDA to meet the GPU memory constraints, but also have >2k trials per batch, we used a low-memory trial sampling algorithm:



Results of NPLDA and E2E models

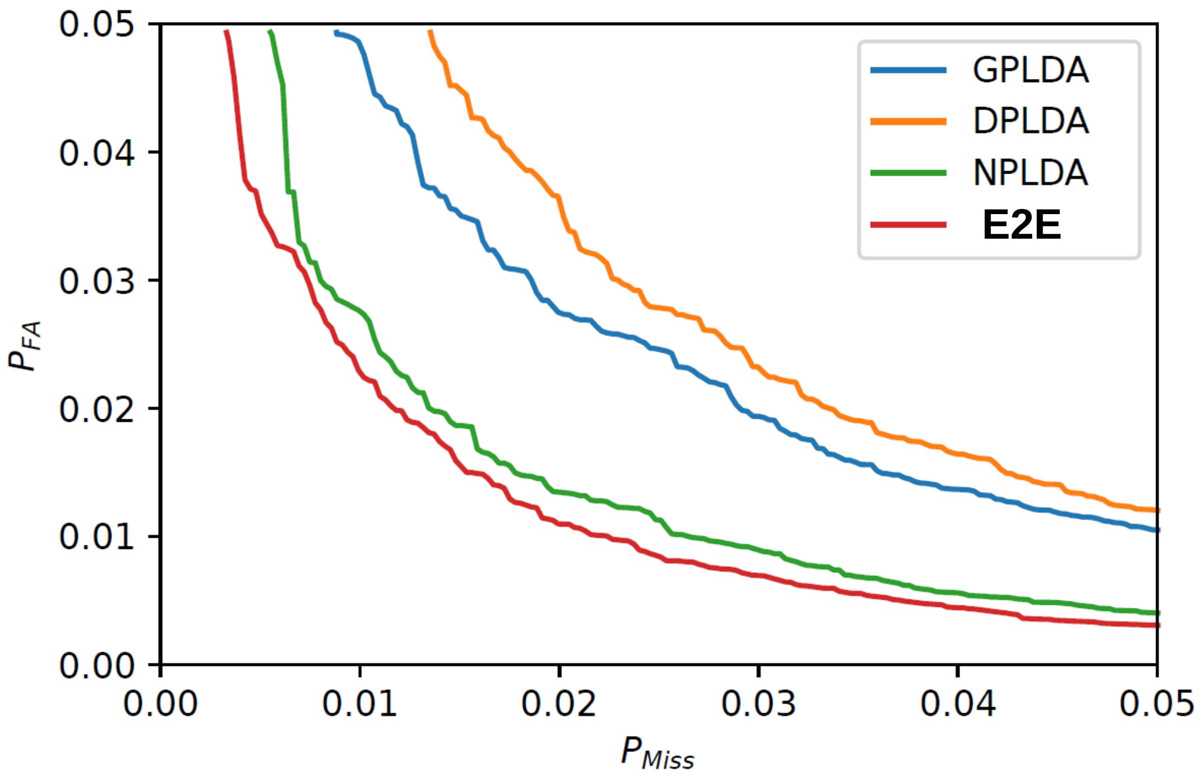
- Results with a factorized TDNN [1] x-vector architecture trained on VoxCeleb 1 & 2 datasets show robust improvements of the model across various test domains.

Model	Init.	SITW-E		VOiCES-D		VOiCES-E		SRE18 D-V		SRE18 E-V	
		EER	C_{Min}	EER	C_{Min}	EER	C_{Min}	EER	C_{Min}	EER	C_{Min}
GPLDA	-	2.68	0.276	2.15	0.268	6.01	0.490	7.41	0.449	15.82	0.610
GB	-	3.47	0.346	2.94	0.315	7.28	0.548	12.76	0.486	17.46	0.647
DPLDA	-	2.90	0.280	2.21	0.273	6.00	0.480	7.82	0.444	13.02	0.553
NPLDA	GPLDA	1.78	0.178	1.49	0.176	5.18	0.432	**6.17**	0.337	12.38	0.467
E2E-NPLDA	GPLDA	1.96	0.177	1.48	0.181	5.45	**0.396**	**6.17**	**0.226**	**12.19**	0.497
E2E-NPLDA	NPLDA	**1.67**	**0.165**	**1.36**	**0.166**	**4.99**	0.407	7.00	0.263	12.68	**0.455**
Relative improvements for E2E-NPLDA over GPLDA in %		38	40	37	38	17	19	17	50	23	25
Relative improvements for E2E-NPLDA over NPLDA in %		6	7	9	6	4	6	-13	19	-2	3

LEAP

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

A look at the Detection Error Tradeoffs



- Detection Error Tradeoff curves are shown for two baseline models (Generative Gaussian PLDA and Discriminative PLDA), Neural PLDA backend, and E2E-NPLDA models.

- Consistent improvements seen across a wide range of thresholds, using many evaluation datasets.

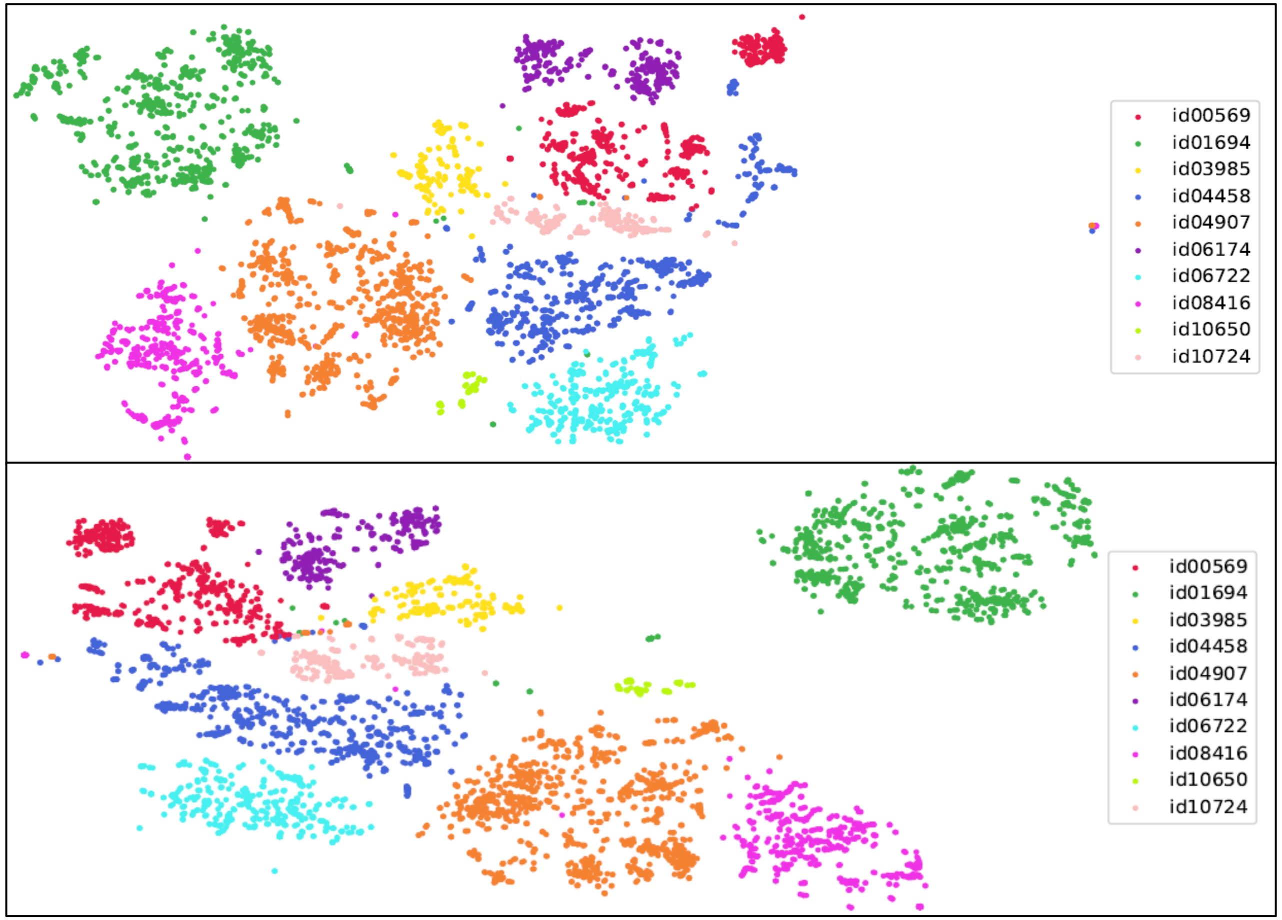
Visualization of Embeddings

Visualization of Embeddings

- t-SNE plots show that the embeddings extracted from the SiamNN models are better suited for the verification task than the original embedding extractor (x-vector model).

Original (x-vector) Embeddings

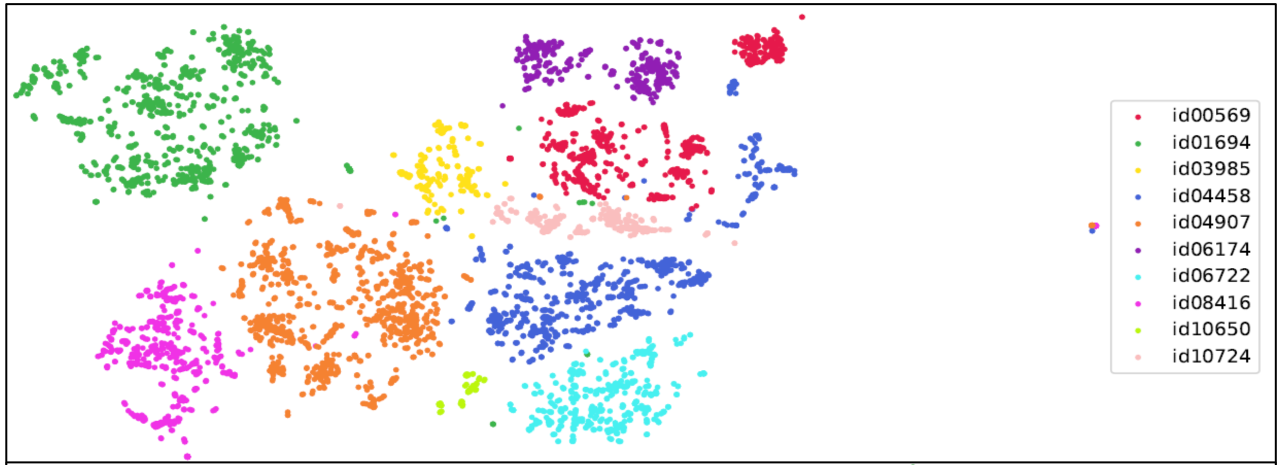
E2E-NPLDA Embeddings



Visualization of Embeddings

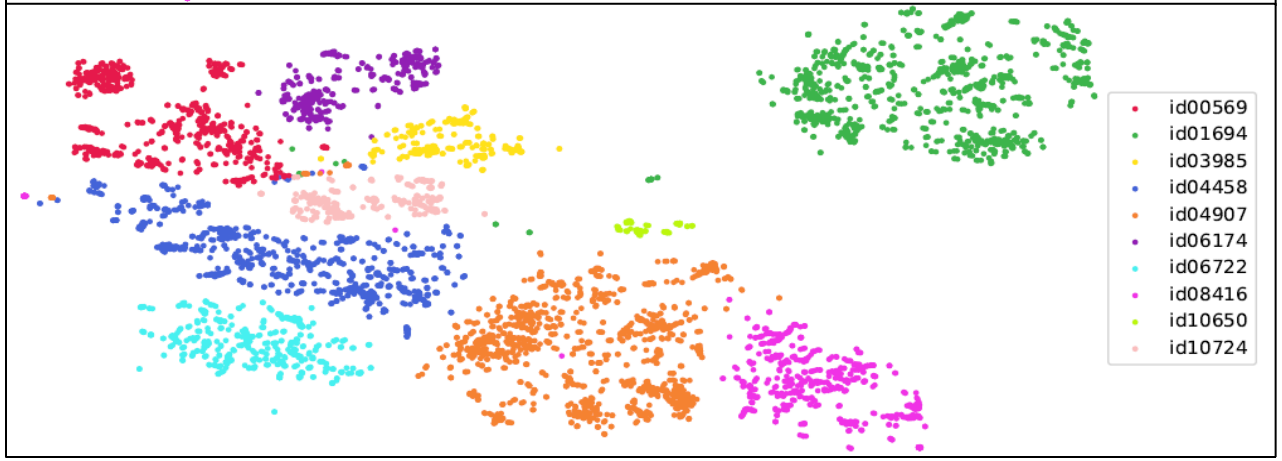
- t-SNE plots show that the embeddings extracted from the SiamNN models are better suited for the verification task than the original embedding extractor (x-vector model).

Original (x-vector) Embeddings



F-Ratio = 8.7

E2E-NPLDA Embeddings



F-Ratio = 15.4

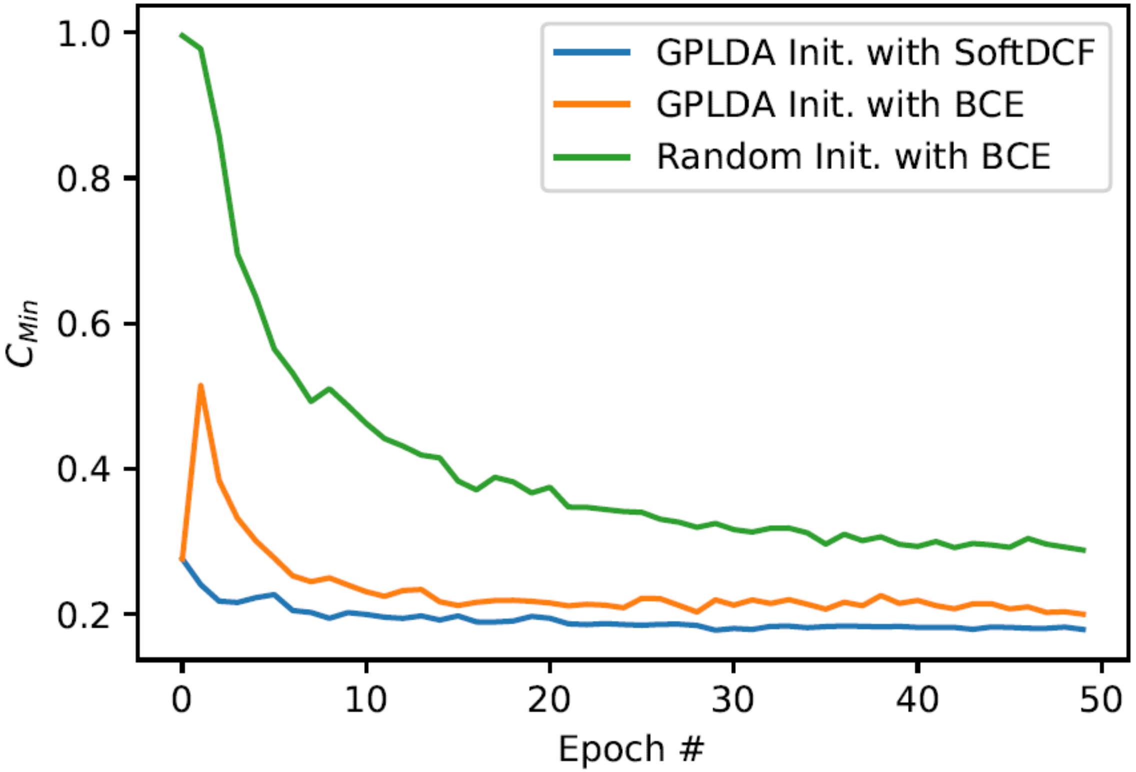
Comparison of Loss functions (NPLDA)

- Comparison of SoftDCF loss with Binary Cross-Entropy and its weighted versions

Embedding Extractor	Init.	Loss Fn.	SITW-E		VOiCES-D		VOiCES-E		SRE18 D-V		SRE18 E-V	
			EER	C_{Min}	EER	C_{Min}	EER	C_{Min}	EER	C_{Min}	EER	C_{Min}
ETDNN	GPLDA	BCE	2.11	0.237	2.52	0.298	6.93	0.585	7.41	0.379	13.02	0.542
		Cllr	2.08	0.241	2.56	0.318	7.30	0.64	8.64	0.449	13.85	0.616
		WCllr	2.19	0.223	2.21	0.26	6.67	0.498	7.41	0.416	**12.38**	0.513
		SoftDCF	2.10	0.209	1.79	0.223	5.69	0.450	7.41	0.377	12.55	0.533
FTDNN	GPLDA	BCE	1.89	0.199	2.10	0.229	6.40	0.538	10.70	**0.337**	**12.38**	0.545
		Cllr	1.95	0.214	2.06	0.246	6.71	0.605	11.11	0.370	13.02	0.561
		WCllr	1.94	0.195	1.95	0.214	5.80	0.474	9.88	0.374	**12.38**	0.500
		SoftDCF	**1.78**	**0.178**	**1.49**	**0.176**	**5.18**	**0.432**	**6.17**	**0.337**	**12.38**	**0.467**

Comparison of Initialization methods

- SoftDCF loss function converges to the best loss values when initialized with GPLDA params.



- With random init, the SoftDCF loss converges to a saddle point where val. $C_{min} = 1$, and all the gradients are 0.

Recent Advances related to our work

- Lots of improvements in the network architecture for speaker verification
  - ResNets, ECAPA-TDNN, Transformer based architectures.
- Exhaustive application of various novel loss functions and their combinations:
  - Classification objectives:
    - Angular Margin (AM) and Additive Angular Margin (AAM) Softmax loss
  - Metric Learning objectives:
    - Angular Prototypical loss

J. S. Chung et. al., "*In defence of metric learning for speaker recognition*" Interspeech 2020

Recent Advances related to our work

Table 4.7: Comparison of the results using the FTDNN models trained on VoxCeleb-2 dataset. The models are tested with VoxCeleb-1 original test set and other evaluation sets. The table also provides comparison with other published works.

Model	Training loss	Back End	Vox1 Test		SITW E		Voices E	
			EER	C_{min}	EER	C_{min}	EER	C_{min}
FTDNN Xvector	Softmax	GPLDA	2.67	0.18	3.2	0.21	6.68	0.38
E2E-NPLDA	Softmax	NPLDA	**2.27**	**0.14**	**2.65**	**0.16**	6.61	**0.35**
VGG-M [1]	Softmax+contrastive	-	5.94	-	-	-	-	-
Resnet34 [1]	Softmax+contrastive	-	4.83	-	-	-	-	-
Resnet50 [1]	Softmax+contrastive	-	3.95	-	-	-	-	-
ResNet34L [2]	Angular	Cosine	2.39	0.17	3.55	0.25	11.1	0.71
ResNet34L [2]	Angular	GPLDA	3.58	0.22	4.16	0.27	10.22	0.54
ResNet34L [2]	Angular	NPLDA	2.46	0.17	3.06	0.21	8.58	0.47

[1] A. Nagrani et. al., "*VoxCeleb: Large scale speaker verification in the wild* ", CSL 2020
[2] J. S. Chung et. al., "*In defence of metric learning for speaker recognition*" Interspeech 2020

Advantages and Disadvantages

- The NPLDA and the corresponding E2E architectures are capable of significantly improving a PLDA based speaker verification model.
- The improvements are consistent over several test sets, over the PLDA model, indicating that it is a robust approach.
- While the softDCF loss helps us achieve the best improvements compared to other loss functions such as the binary cross-entropy and its weighted versions, it is highly sensitive to factors such as batch size, learning rate, and the warping factor α.
- The softDCF loss requires the parameters to be initialized with the Generative PLDA parameters, and not suitable for a fully end-to-end training of the network.

Concluding remarks…

Summary and Contributions

Summary and Contributions

Summary and Contributions

- Speaker and Language – Two very important, independent attributes of speech.

Summary and Contributions

- Speaker and Language – Two very important, independent attributes of speech.
- S/L Recognition – Have similar problems statements – Classification, Detection or Verification, and Diarization.

Summary and Contributions

- Speaker and Language – Two very important, independent attributes of speech.
- S/L Recognition – Have similar problems statements – Classification, Detection or Verification, and Diarization.
- Two major directions pursued:

Summary and Contributions

- Speaker and Language – Two very important, independent attributes of speech.
- S/L Recognition – Have similar problems statements – Classification, Detection or Verification, and Diarization.
- Two major directions pursued:
  - Supervised i-vector (S-vector) modeling for Language/Accent Recognition.
  - PLDA inspired Siamese neural networks for Speaker Verification.

Summary and Contributions

- Speaker and Language – Two very important, independent attributes of speech.
- S/L Recognition – Have similar problems statements – Classification, Detection or Verification, and Diarization.
- Two major directions pursued:
  - Supervised i-vector (S-vector) modeling for Language/Accent Recognition.
    - Algorithms & Derivations, Embedding Extraction methods.
  - PLDA inspired Siamese neural networks for Speaker Verification.

Summary and Contributions

- Speaker and Language – Two very important, independent attributes of speech.
- S/L Recognition – Have similar problems statements – Classification, Detection or Verification, and Diarization.
- Two major directions pursued:
  - Supervised i-vector (S-vector) modeling for Language/Accent Recognition.
    - Algorithms & Derivations, Embedding Extraction methods.
  - PLDA inspired Siamese neural networks for Speaker Verification.
    - NPLDA backend architecture, E2E training, loss functions and their detailed comparison.

Future Perspectives

- Neural network based generative models for speaker and language embedding extraction for recognition.

- Neural Network approaches for language recognition, particularly for lang diarization.

- Self-supervised Learning for speaker and language recognition.

- Architectures for *speaker detection* in conversational settings.

    - *Cross Attention* between enrollment and test segments that can better model the speaker in language/accent/emotion mismatched cases, and overlapped speech.

Publications based on this thesis

- Journal articles

  - S. Ramoji, S. Ganapathy, “Supervised I-vector modeling for language and accent recognition,” Computer Speech \& Language 60, (2020): p.101030.

  - S. Ramoji, P. Krishnan, S. Ganapathy, “PLDA inspired Siamese networks for speaker verification,” Computer Speech \& Language 76, (2022): p.101383.

Publications based on this thesis

- Conference Publications
  - S. Ramoji, S. Ganapathy, “Supervised I-vector Modeling-Theory and Applications,” in Proc. Interspeech (2018): p.1091-1095.
  - S. Ramoji et. al., “The LEAP speaker recognition system for NIST SRE 2018 challenge,” in Proc. ICASSP (2019): p.5771-5775.
  - S. Ramoji, P. Krishnan, S. Ganapathy, “NPLDA: A Deep Neural PLDA Model for Speaker Verification,” in Proc. Odyssey (2020): p.202-209.
  - S. Ramoji, P. Krishnan, S. Ganapathy, “LEAP System for SRE 2019 CTS Challenge - Improvements and Error Analysis,” in Proc. Odyssey (2020): p.281-288.
  - S. Ramoji, P. Krishnan, S. Ganapathy, “Neural PLDA Modeling for End-to-End Speaker Verification,” Proc. Interspeech (2020): p.4333-4337

Acknowledgements

Acknowledgements

- Firstly, I thank Dr. Sriram Ganapathy for his guidance, and most importantly, for being such an understanding and empathic advisor.

Acknowledgements

- Firstly, I thank Dr. Sriram Ganapathy for his guidance, and most importantly, for being such an understanding and empathic advisor.

- I thank my comprehensive examination panel – Dr T. V. Sreenivas, Dr. Aditya Gopalan and Dr. P. S. Sastry, and my thesis examiners Dr. Luciana Ferrer and Dr. Rohit Sinha.

Acknowledgements

- Firstly, I thank Dr. Sriram Ganapathy for his guidance, and most importantly, for being such an understanding and empathic advisor.

- I thank my comprehensive examination panel – Dr T. V. Sreenivas, Dr. Aditya Gopalan and Dr. P. S. Sastry, and my thesis examiners Dr. Luciana Ferrer and Dr. Rohit Sinha.

- I thank all my co-authors for their contributions in various Speaker and Language Recognition Challenges we participated in: Bharath Padi, Vaishnavi Y, Anand Mohan, Satish Kumar, Anmol Bhatia, Bhargavaram Mysore, Prachi Singh, and Prashant Krishnan.

Acknowledgements

- Firstly, I thank Dr. Sriram Ganapathy for his guidance, and most importantly, for being such an understanding and empathic advisor.

- I thank my comprehensive examination panel – Dr T. V. Sreenivas, Dr. Aditya Gopalan and Dr. P. S. Sastry, and my thesis examiners Dr. Luciana Ferrer and Dr. Rohit Sinha.

- I thank all my co-authors for their contributions in various Speaker and Language Recognition Challenges we participated in: Bharath Padi, Vaishnavi Y, Anand Mohan, Satish Kumar, Anmol Bhatia, Bhargavaram Mysore, Prachi Singh, and Prashant Krishnan.

- Special mentions to Bharath Padi and Anand Mohan – The ones who really helped me with tricky tools such as Kaldi.

Acknowledgements

- Firstly, I thank Dr. Sriram Ganapathy for his guidance, and most importantly, for being such an understanding and empathic advisor.

- I thank my comprehensive examination panel – Dr T. V. Sreenivas, Dr. Aditya Gopalan and Dr. P. S. Sastry, and my thesis examiners Dr. Luciana Ferrer and Dr. Rohit Sinha.

- I thank all my co-authors for their contributions in various Speaker and Language Recognition Challenges we participated in: Bharath Padi, Vaishnavi Y, Anand Mohan, Satish Kumar, Anmol Bhatia, Bhargavaram Mysore, Prachi Singh, and Prashant Krishnan.

- Special mentions to Bharath Padi and Anand Mohan – The ones who really helped me with tricky tools such as Kaldi.

- Another special mention to Prashant Krishnan, the one who I worked with for the longest – I would like to thank him for helping me run various experiments, organize all the results, and for all the technical and random non-technical discussions we had.

Acknowledgements

Acknowledgements

- I also thank Sri Garimella for giving me the internship opportunity at Amazon Alexa, Harish Arsikere for mentoring me during this internship, and all my other colleagues in SriGar Team, Amazon, Bangalore.

Acknowledgements

- I also thank Sri Garimella for giving me the internship opportunity at Amazon Alexa, Harish Arsikere for mentoring me during this internship, and all my other colleagues in SriGar Team, Amazon, Bangalore.

- Special mentions to Sriram Ganapathy, Neeraj Sharma, Purvi Agrawal, Akshara Soman, Anurenjan, Prachi Singh and Jaswanth Reddy for all the memorable trips and fun we had together that made this journey memorable.

Acknowledgements

- I also thank Sri Garimella for giving me the internship opportunity at Amazon Alexa, Harish Arsikere for mentoring me during this internship, and all my other colleagues in SriGar Team, Amazon, Bangalore.

- Special mentions to Sriram Ganapathy, Neeraj Sharma, Purvi Agrawal, Akshara Soman, Anurenjan, Prachi Singh and Jaswanth Reddy for all the memorable trips and fun we had together that made this journey memorable.

- Penultimately, I would like to thank all my friends, the present and past members of LEAP lab, however brief or long our interactions were for being a part of this journey and making it memorable.

Acknowledgements

- I also thank Sri Garimella for giving me the internship opportunity at Amazon Alexa, Harish Arsikere for mentoring me during this internship, and all my other colleagues in SriGar Team, Amazon, Bangalore.

- Special mentions to Sriram Ganapathy, Neeraj Sharma, Purvi Agrawal, Akshara Soman, Anurenjan, Prachi Singh and Jaswanth Reddy for all the memorable trips and fun we had together that made this journey memorable.

- Penultimately, I would like to thank all my friends, the present and past members of LEAP lab, however brief or long our interactions were for being a part of this journey and making it memorable.

- Finally, and most importantly, I extend my sincere thanks to my parents, whose love and support can simply not be measured. I dedicate my PhD thesis to them.

Thank you for your attention !