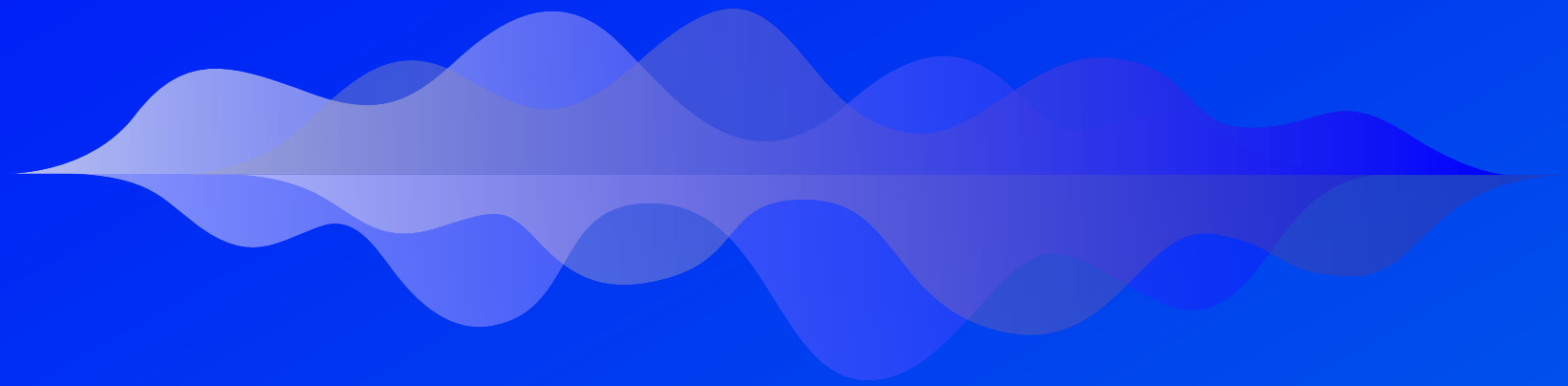
Speaker Recognition

x-vectors

Ali Majlesi 1400/5/17

Outline

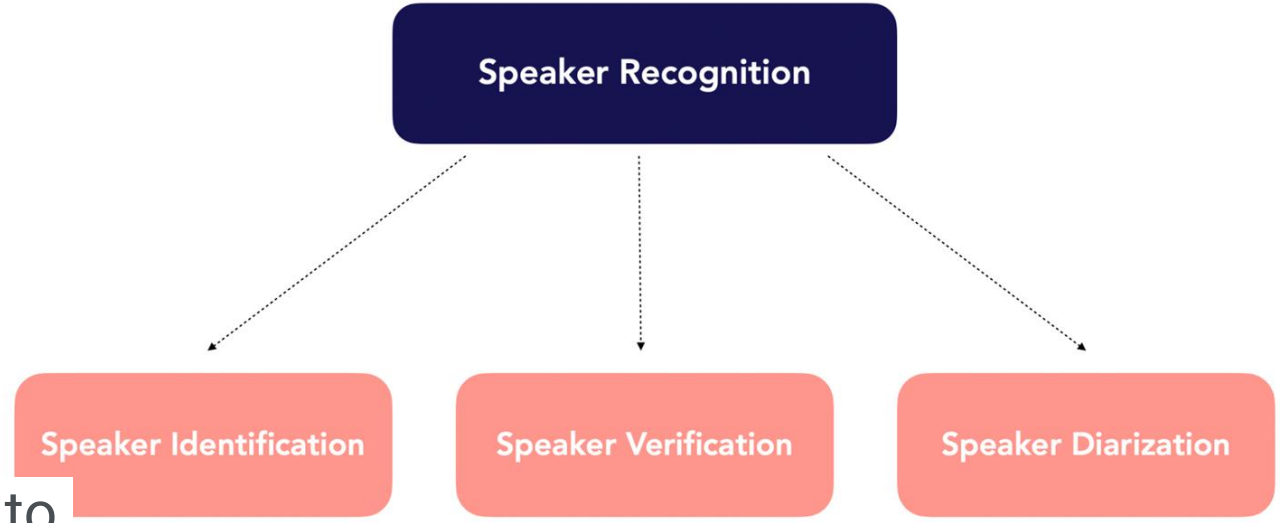
- Applications of speaker recognition
- Neural Network method (X-vector)
- Compare with previous method (GMM-UBM)



Speaker recognition

Speaker biometrics/recognition is split into:

- *Speaker identification*: determine an unknown speaker's identify among a group of speakers.
- *Speaker verification*: verify the claimed identity of a person through speech signal.
- *Speaker diarization*: partition an audio input stream into segments according to the speaker identity.



Why do we need Speaker Recognition?

- Block or grant the access to a service (e.g. bank)
- Activate a voice assistant on your voice only (e.g. Hey Siri)
- Criminal investigations on intercepted telephone calls
- ....

How does speaker verification work?

Speaker verification can be:

- **Text dependent:** must say for example Hey Siri to activate a service
- **Text independent:** your voice is enough to launch the service

The high-level idea is very simple. You usually have 3 sets of data:

- Enrollment data: Contains recording of the speakers to recognize.
- Development data: Used for adaptation. (Optional)
- Test data: Where you must predict the identities.

JOHNS HOPKINS UNIVERSITY

human language technology center of excellence

X-vectors: Neural Embeddings (IS 2017, ICASSP 2018)

Deep Neural Network Embeddings for Text-Independent Speaker Verification

David Snyder, Daniel Garcia-Romero, Daniel Povey, Sanjeev Khudanpur

Center for Language and Speech Processing & Human Language Technology Center of Excellence,
The Johns Hopkins University, USA

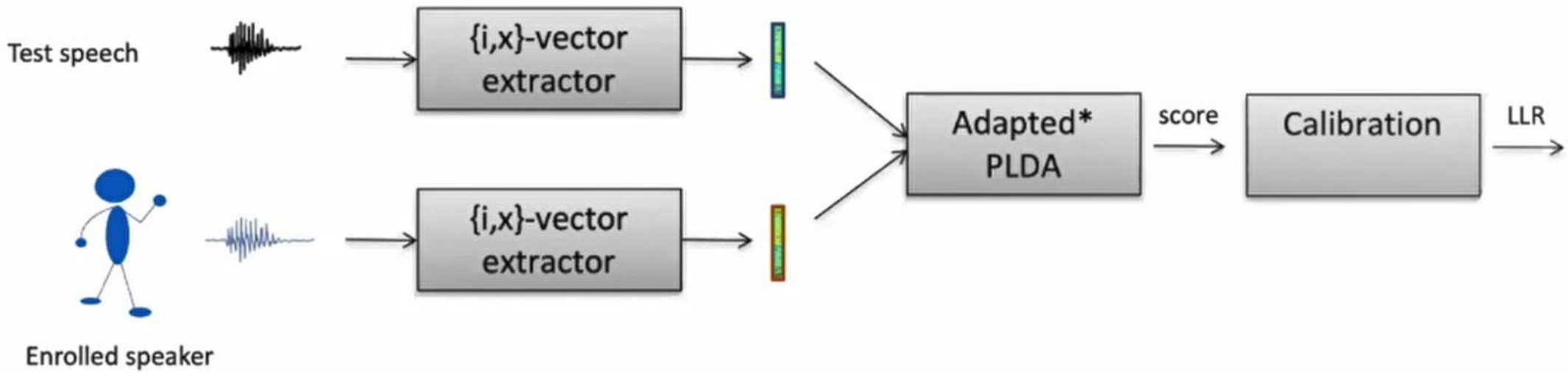
X-VECTORS: ROBUST DNN EMBEDDINGS FOR SPEAKER RECOGNITION

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, Sanjeev Khudanpur

Center for Language and Speech Processing & Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA

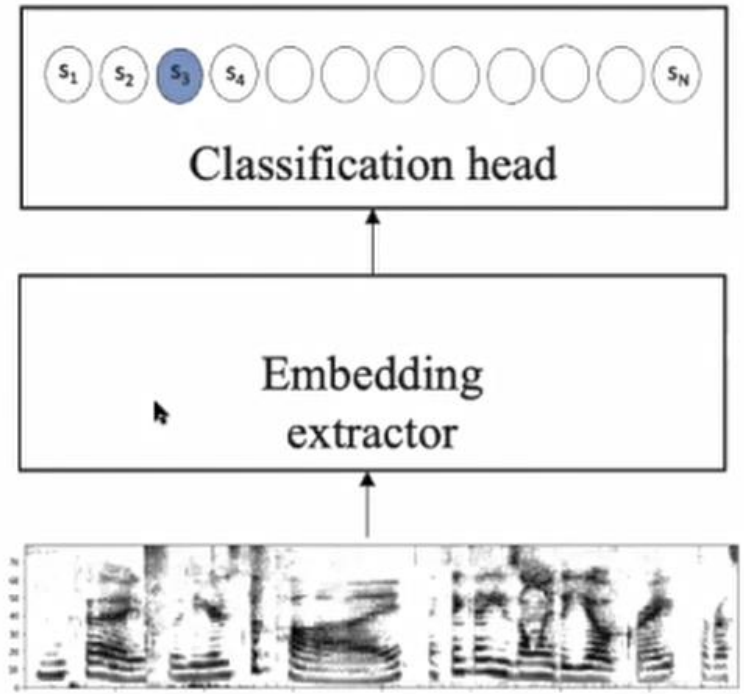
NIST SRE16 Cantonese (conversational telephone speech)

- Domain mismatch between training data and deployment environment:
  - Training data (extractor, PLDA, calibration):
    - Out of-domain (mostly English collected in US)
    - Small set of unlabeled in-domain data (used for PLDA adaptation)
  - Enroll and test segments: Cantonese collected in China

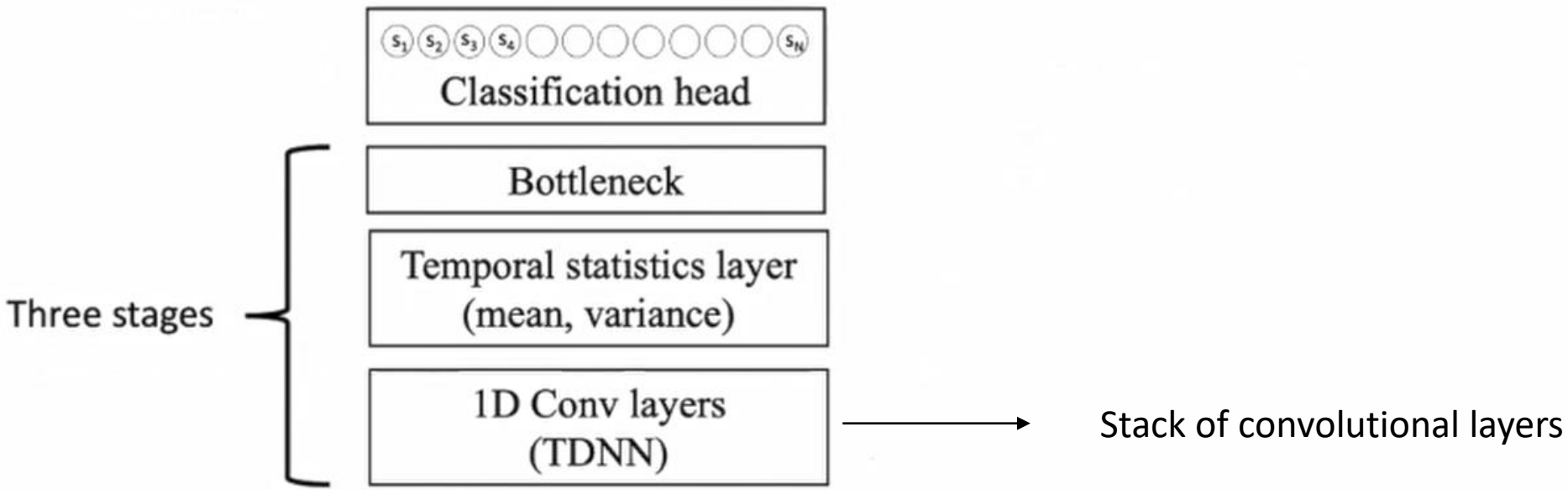


X-vector system overview

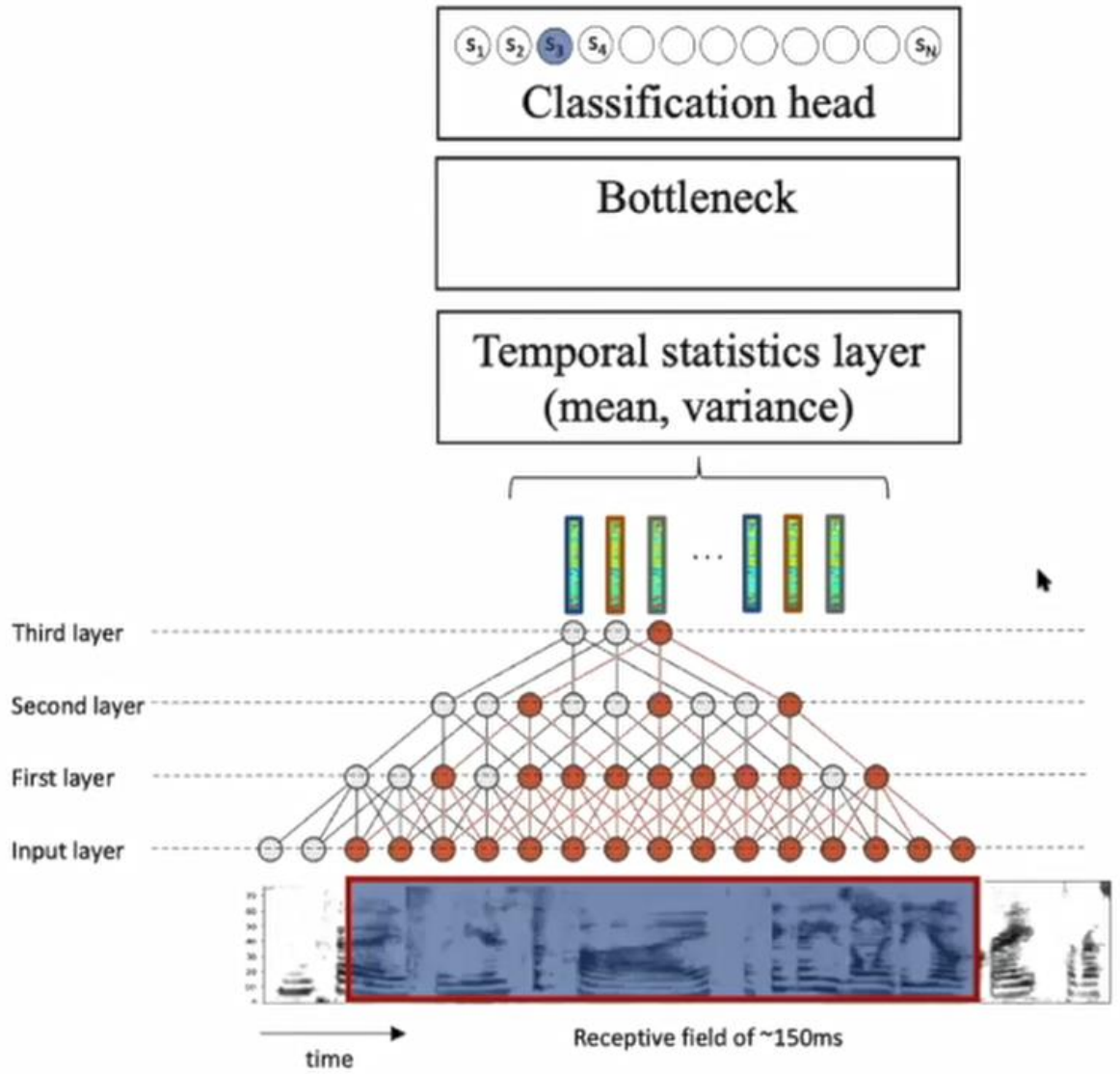
- Two modules:
  - Embedding extractor (x-vectors)
    - Summarize speaker information
  - Classification head
    - Surrogate classification task to produce discriminative embeddings
    - We use categorical cross-entropy loss



X-vector extractor



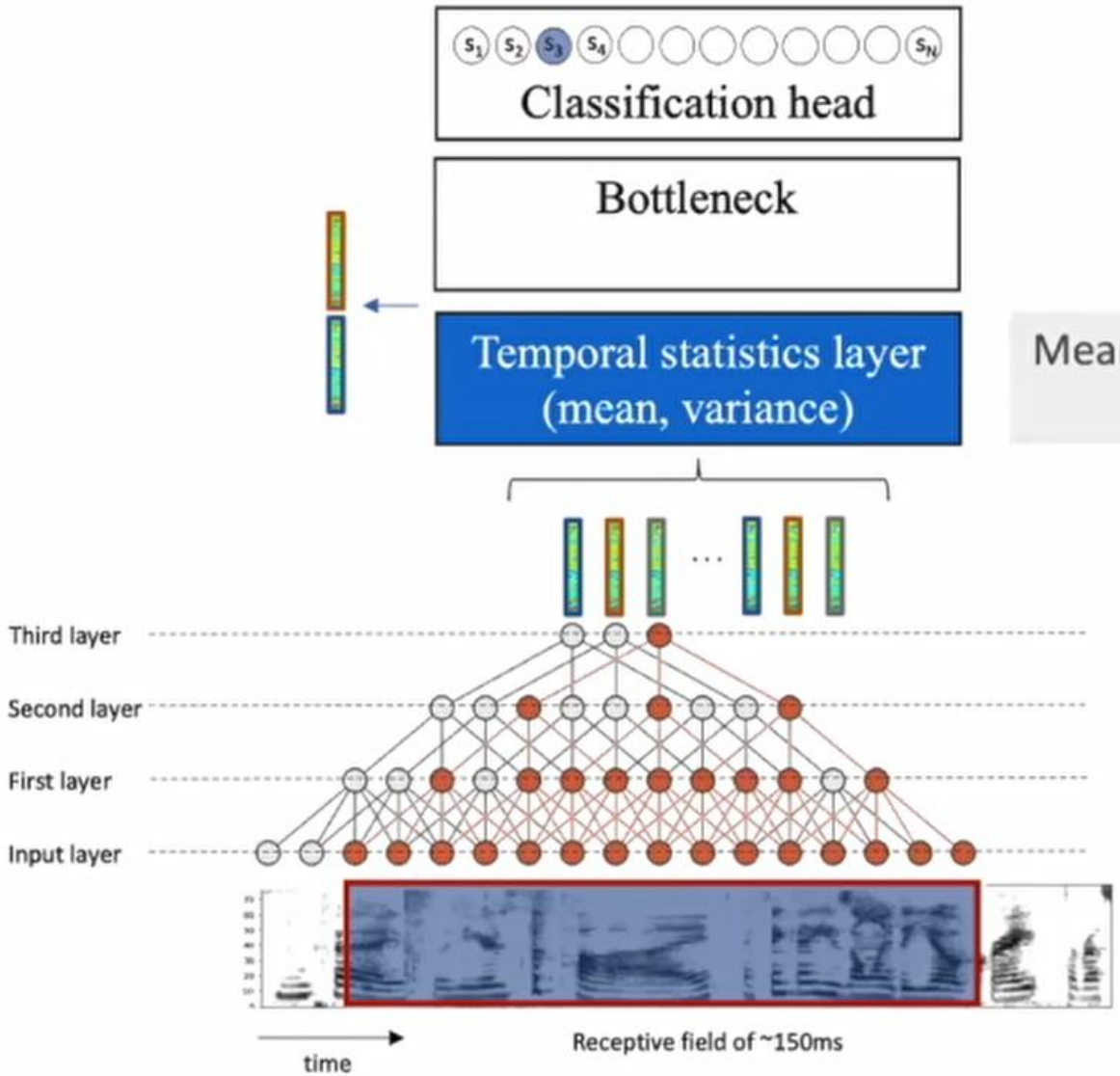
1D-conv stack (TDNN)



Sequence of higher level representations to facilitate speaker discrimination

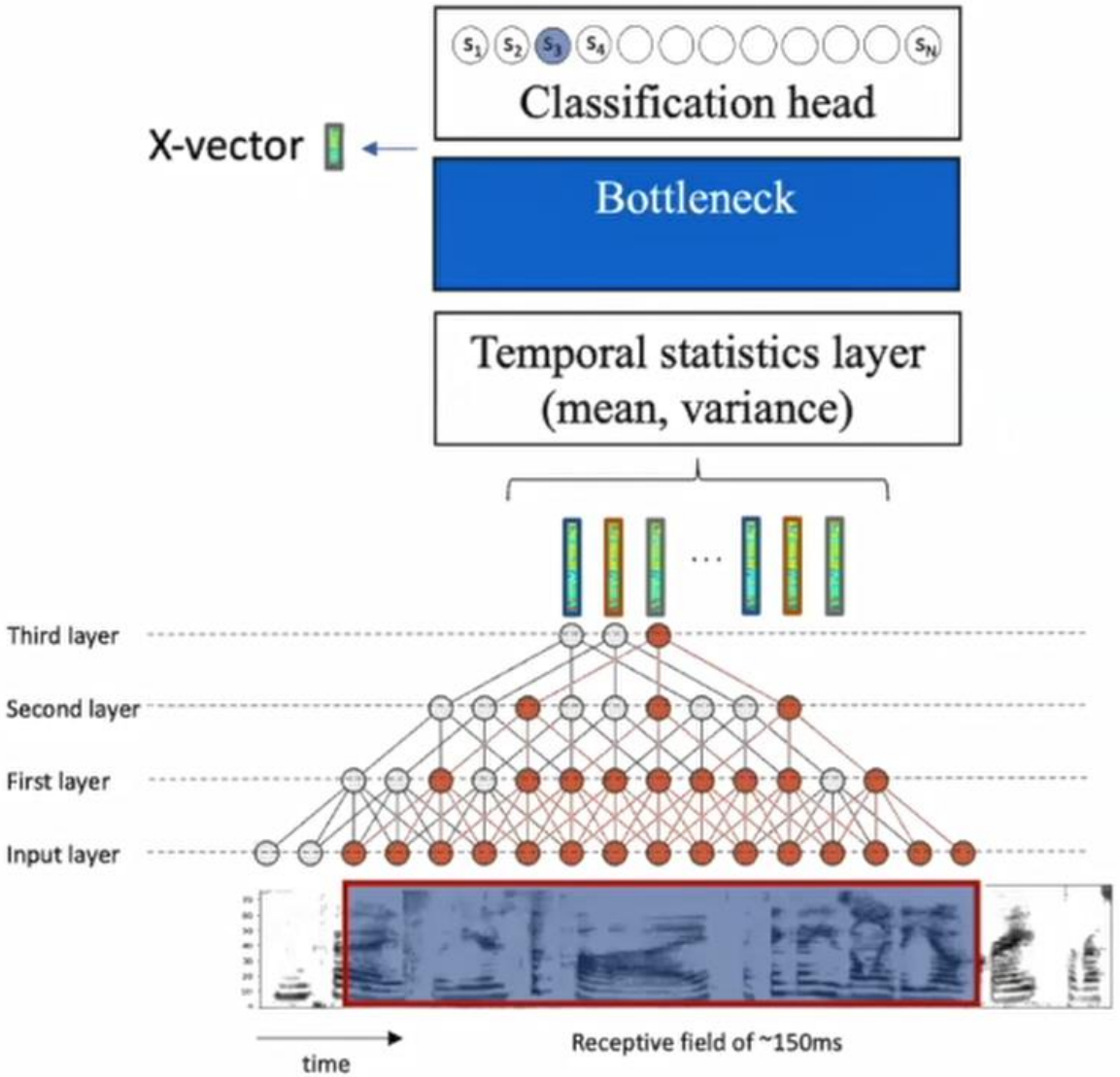
Sequence of mel-filterbank energies

Temporal pooling



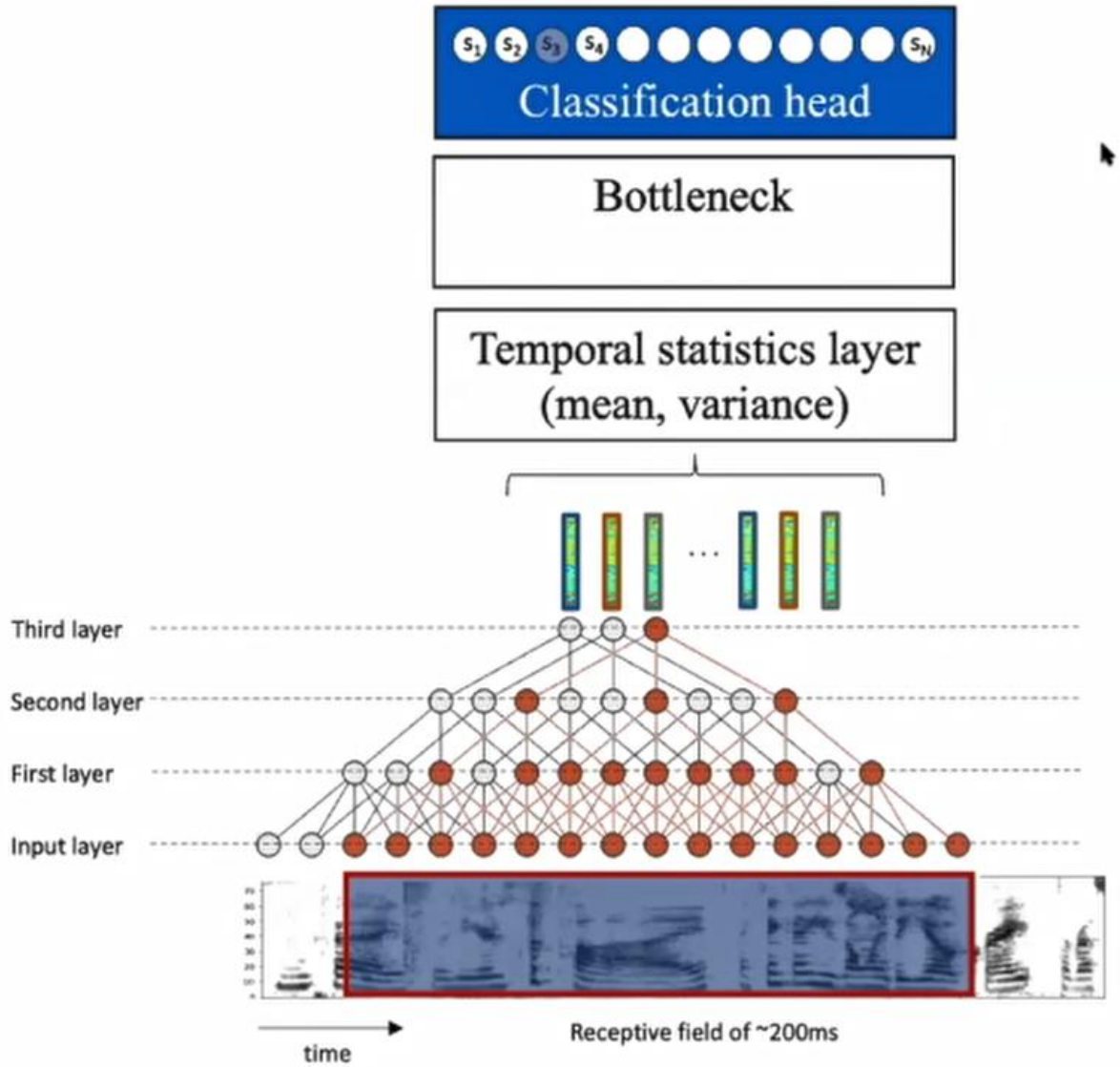
Mean and variance summary of each dimension:
fixed-length vector

Bottleneck



Dimensionality reduction:
From 3-5k -> 256 or 512

Classification head



Categorical cross-entropy loss

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{e^{\mathbf{W}_{y_i}\mathbf{x}+b_{y_i}}}{\sum_j e^{\mathbf{W}_j\mathbf{x}+b_j}}$$

Summary of Network

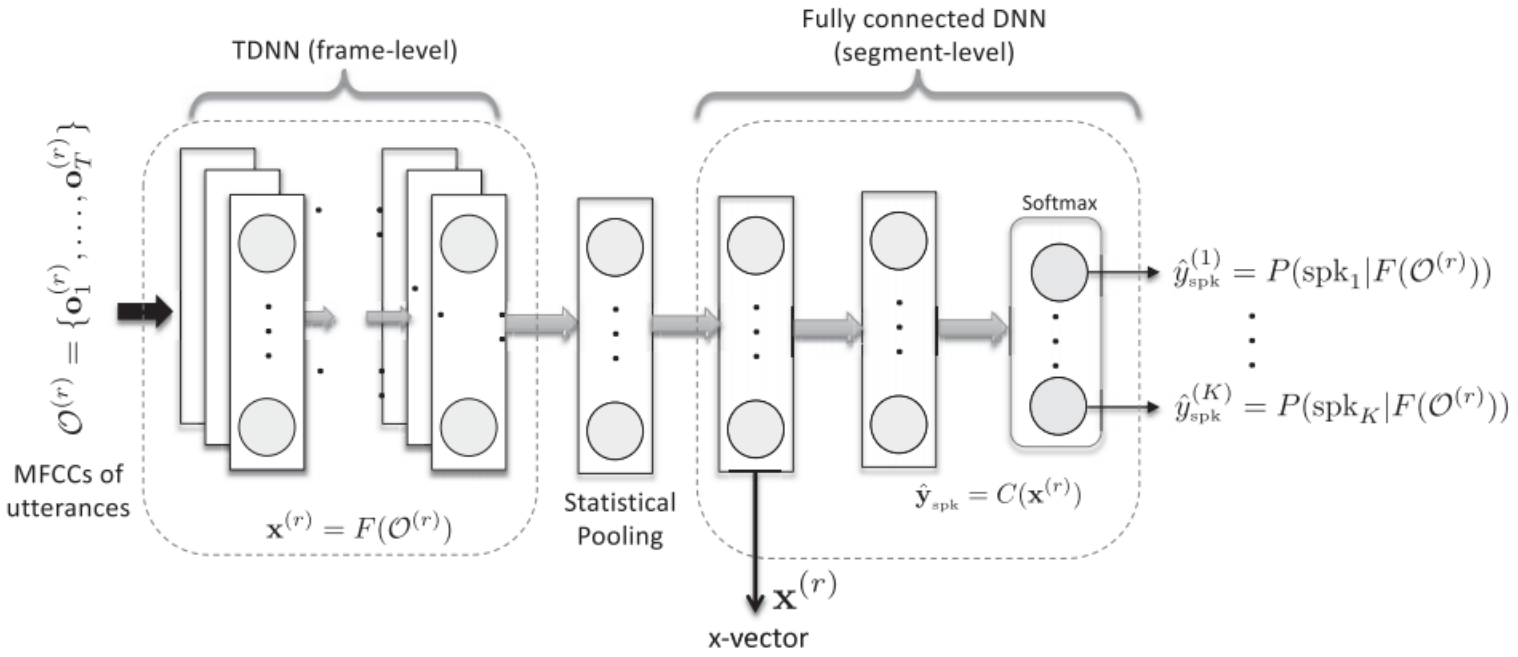
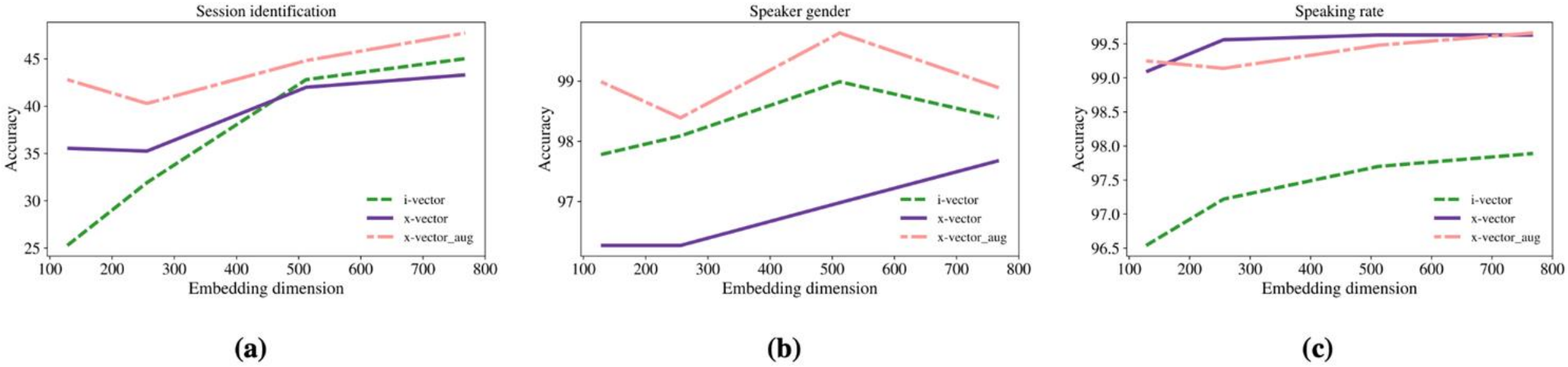


Figure 5.4 Structure of an x-vector extractor. The TDNN aims to capture the temporal information of contextual acoustic frames and the pooling layer aggregates the temporal information to form the utterance-level representation called the x-vector.

What do these embeddings capture?

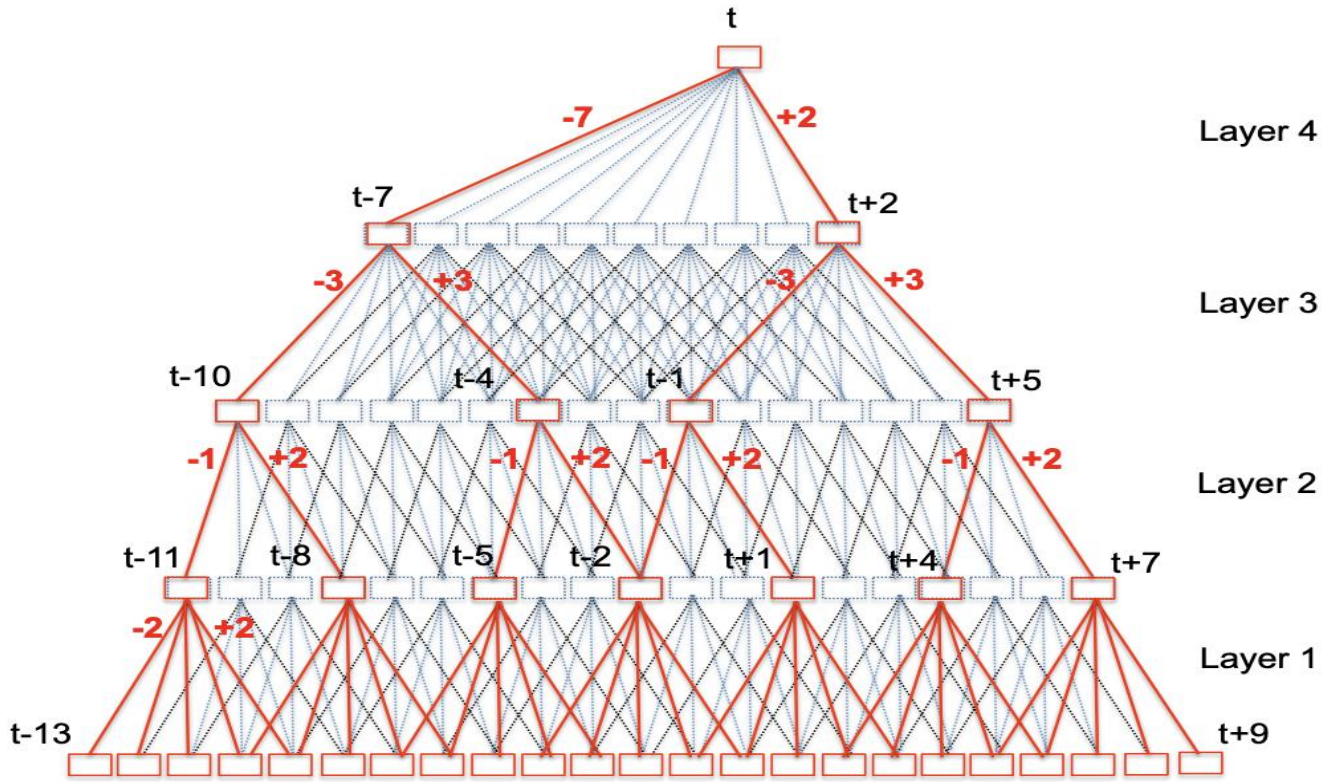
Embeddings have been explored to solve other tasks and see what they capture:



Read the paper here

Training a DNN to classify speakers - X vectors

TDNN : Time Domain Neural Networks



TDNN

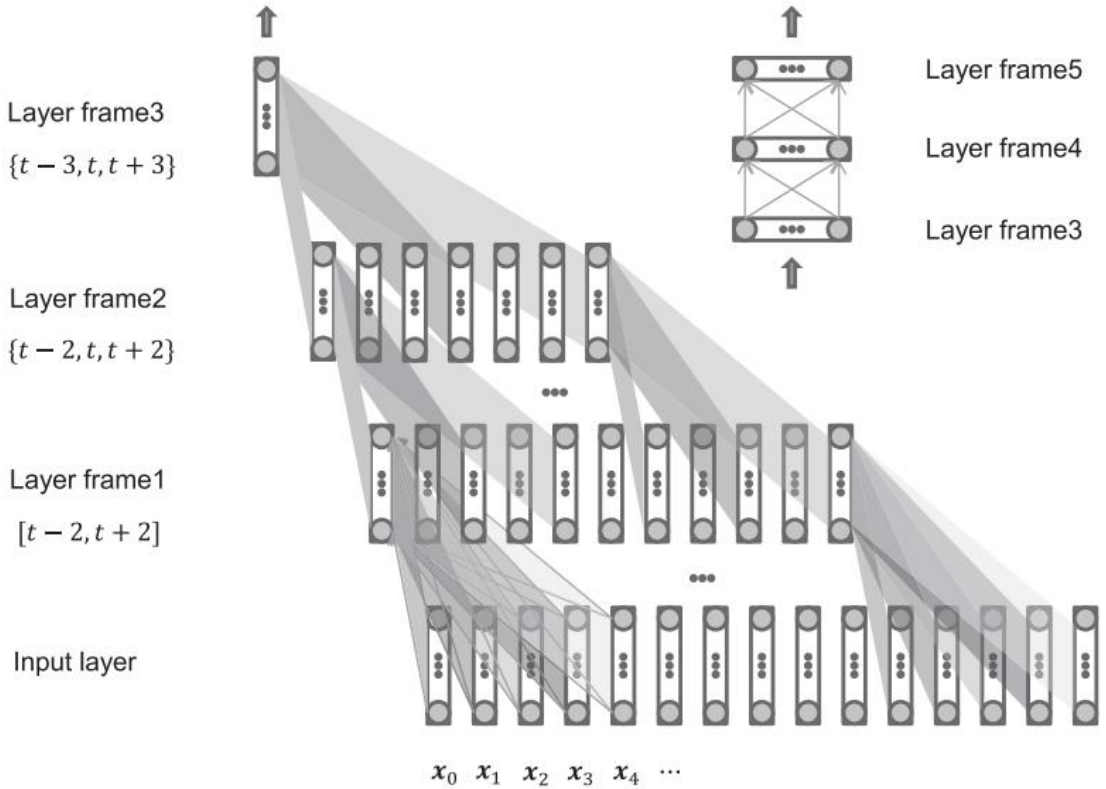
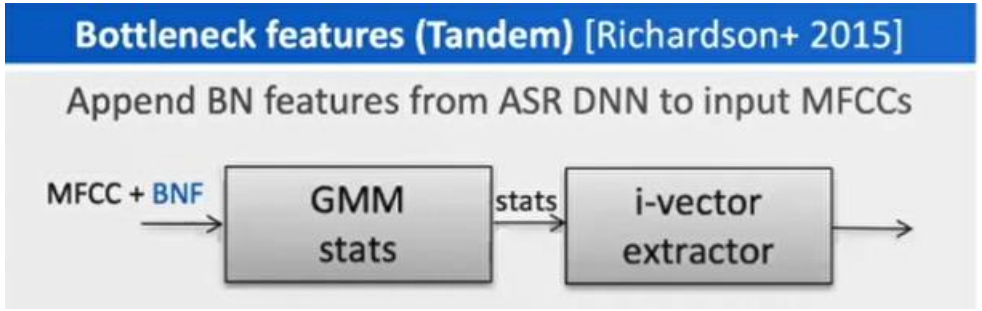
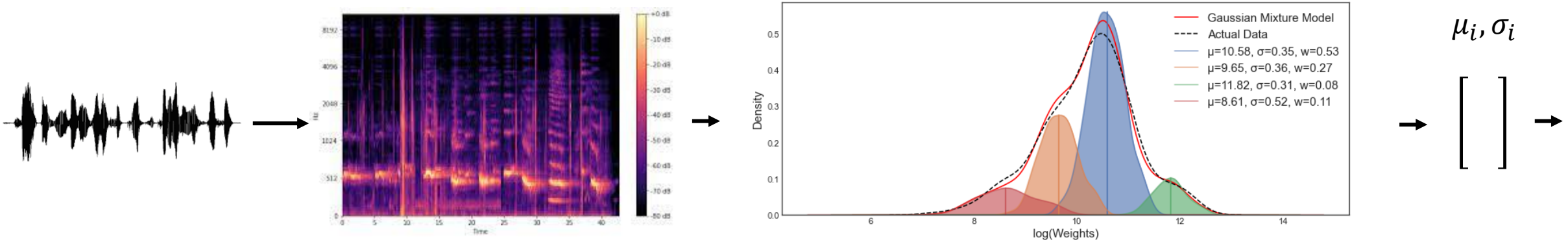


Figure 5.6 Architecture of the TDNN in the x-vector extractor. At the input layer, the vertical bars represent acoustic frames. In the hidden layers, the vertical bars comprises neurons with connections to contextual nodes in the layer below. (Courtesy of Youzhi Tu)

Previous Algorithms

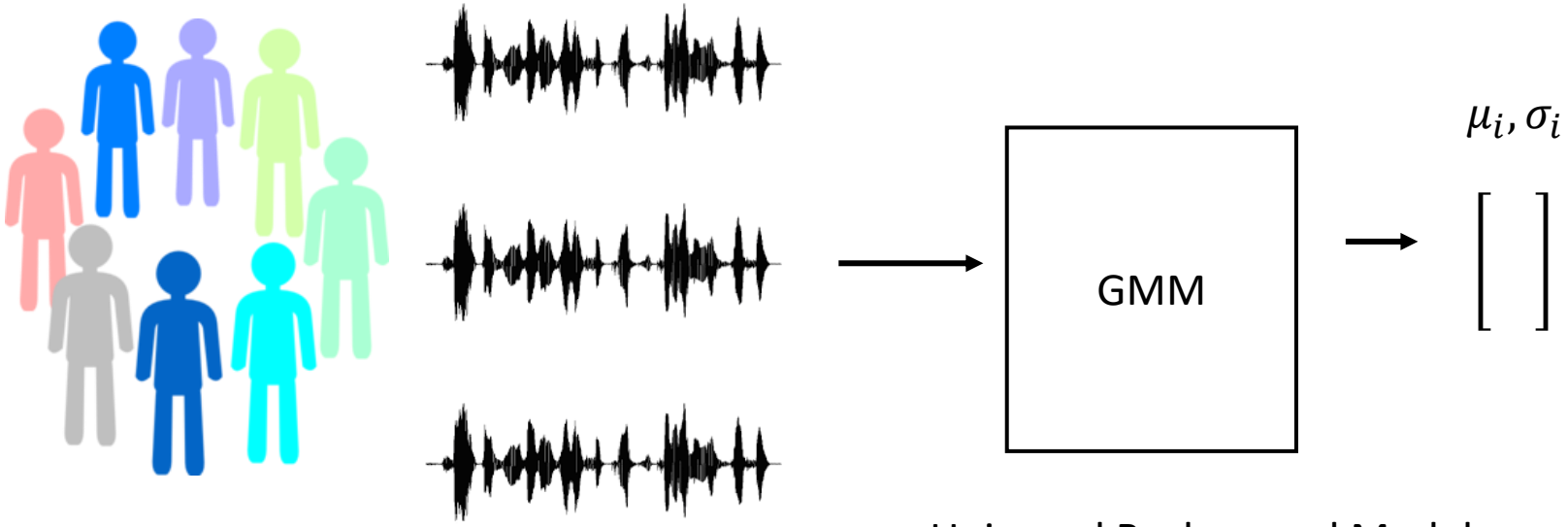




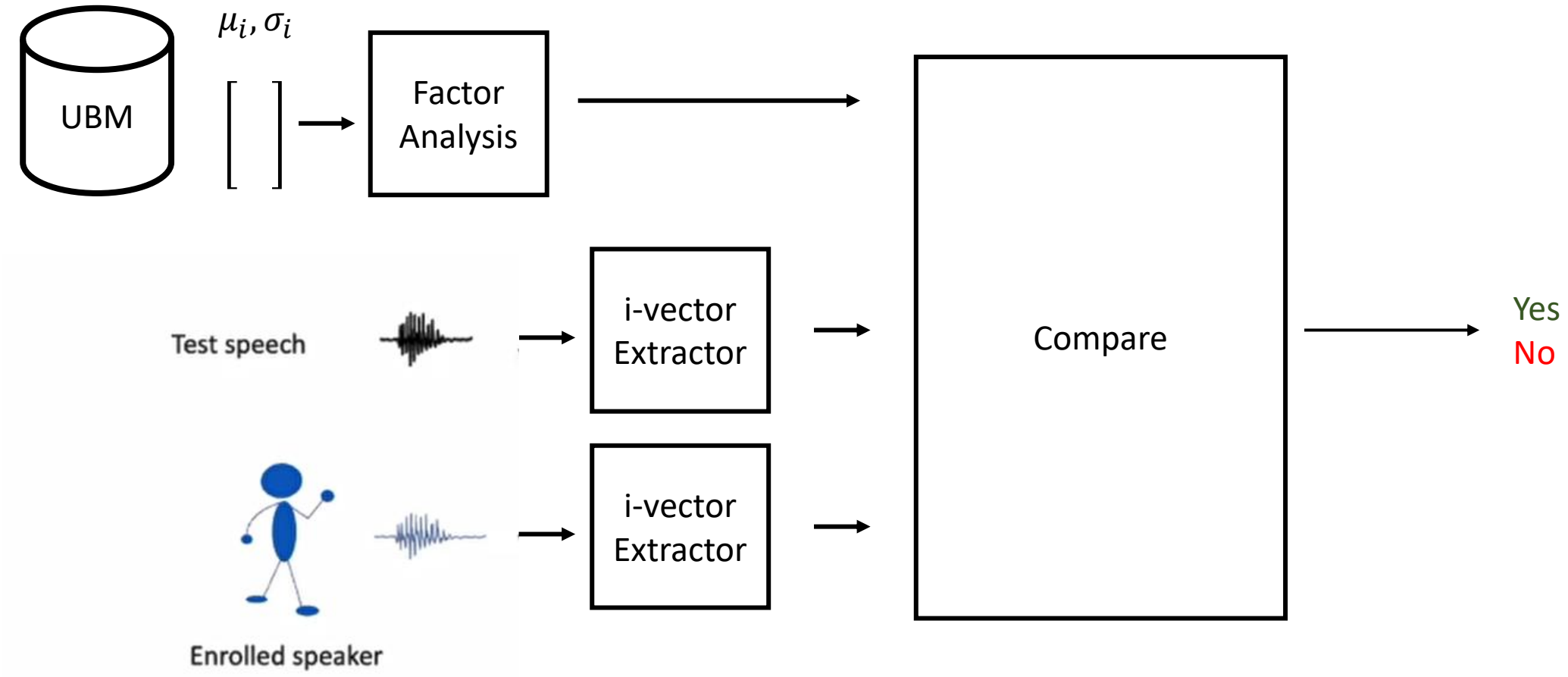
μ_i, σ_i

μ_i, σ_i → $Factor\ Analysis$ → $i-vector$

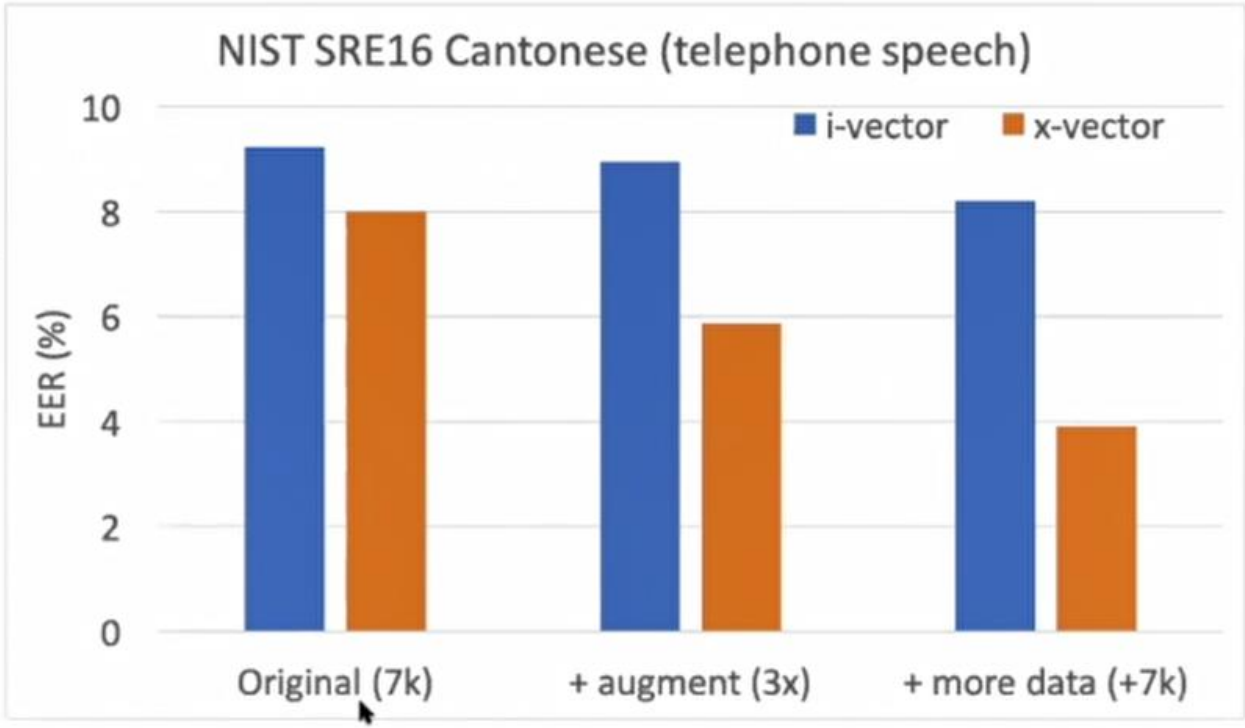
i-vector



i-vector

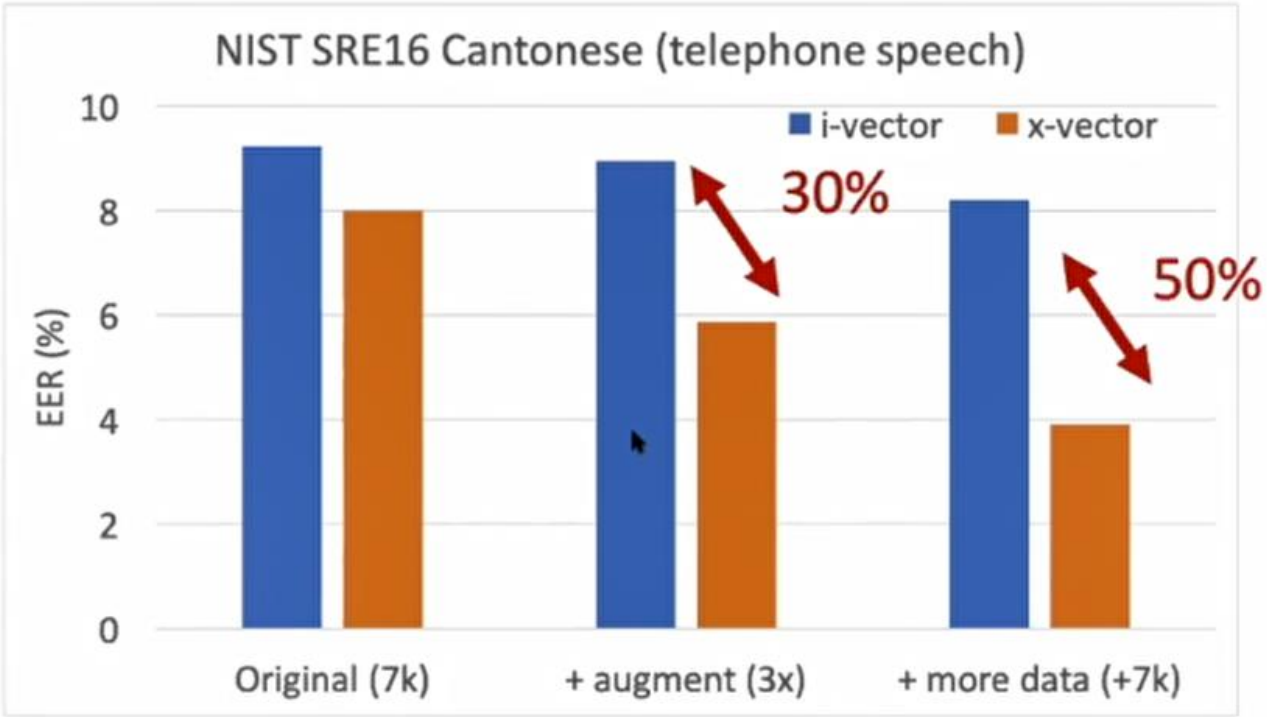

UBM
μ_i, σ_i
Factor Analysis
Test speech
i-vector Extractor
i-vector Extractor
Enrolled speaker
Compare
Yes
No

Results



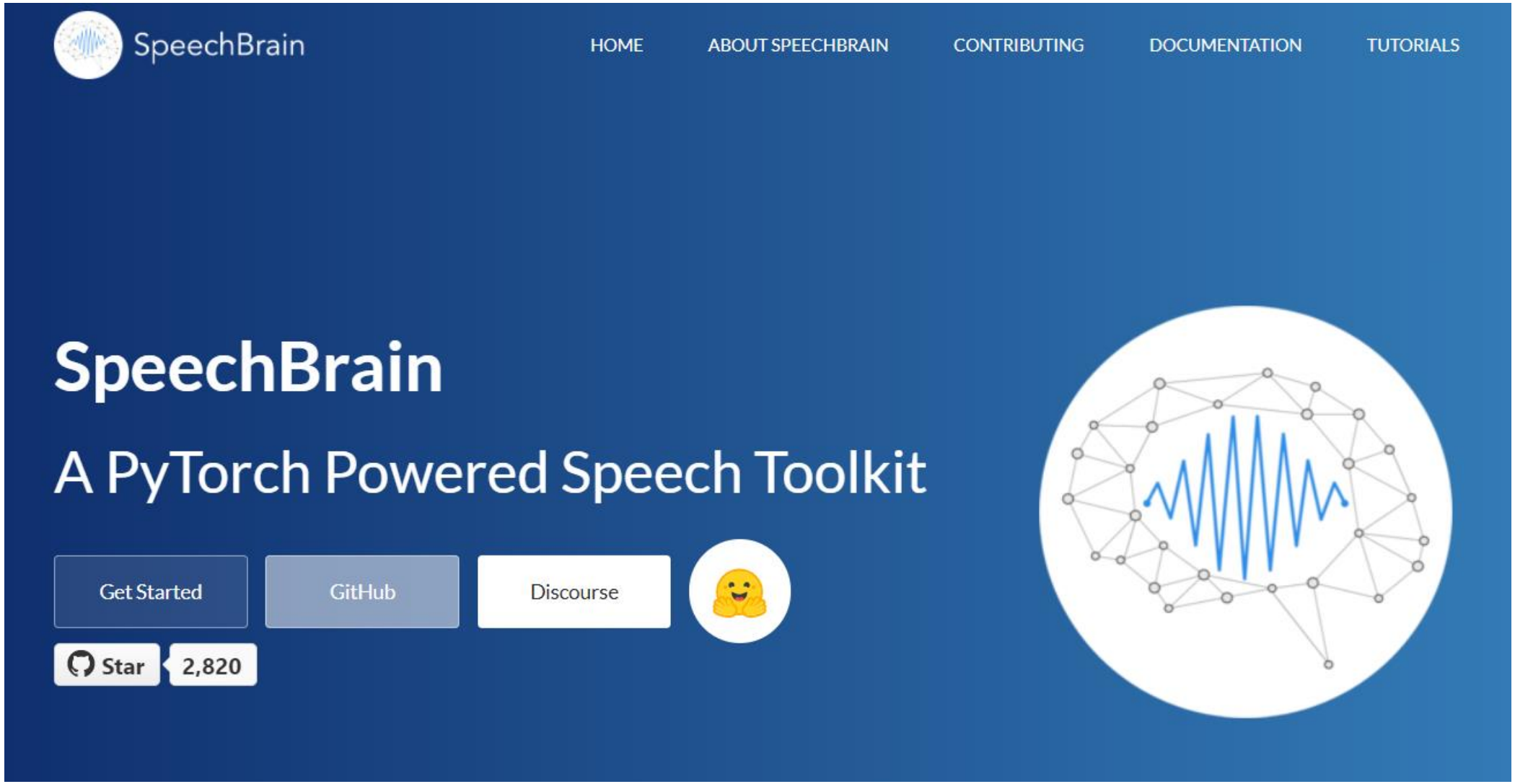
- Apples-to-apple comparison (replacing the i-vector with the x-vector)

Results



- Apples-to-apple comparison (replacing the i-vector with the x-vector)
- Augmentation is critical for x-vector system (not so much for unsupervised i-vector extractor)
  - The supervision of x-vector training results in better generalization
- X-vectors scale better with more data

Speech Brain



How to extract X-vectors with SpeechBrain?

```python
import torchaudio
from speechbrain.pretrained import SpeakerRecognition

verif_model = SpeakerRecognition.from_hparams(source="speechbrain/spkrec-ecapa-voxceleb",
savedir="pretrained_models/spkrec-ecapa-voxceleb")
signal, fs = torchaudio.load("test.wav")
embeddings = verif_model.encode_batch(signal)
```

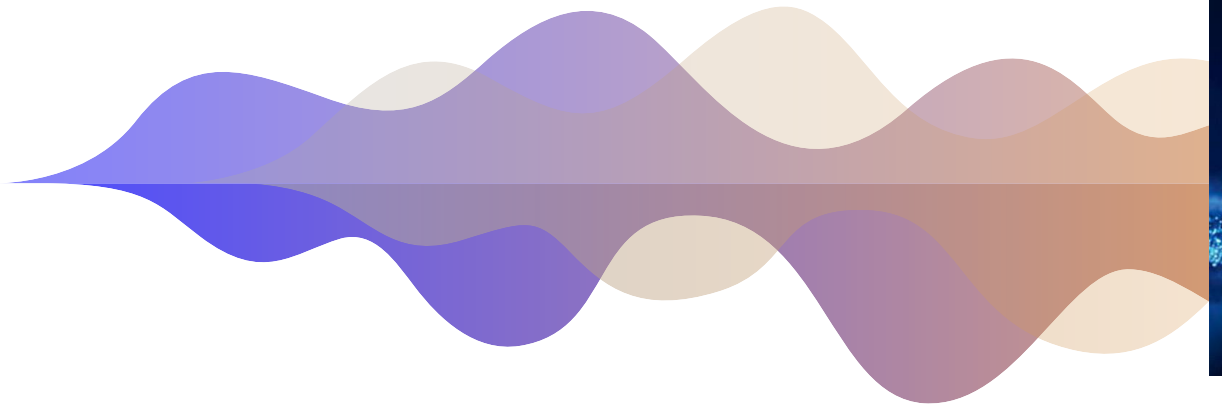
```
torch.Size([2, 1, 192])
```

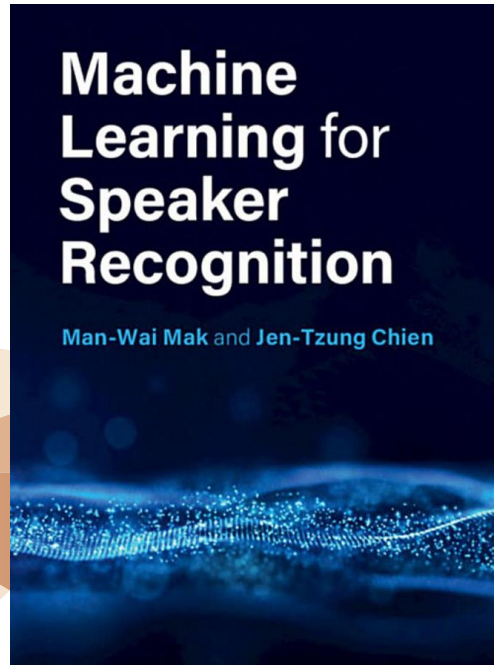
The pipeline is SpeechBrain

```
# Load model
verif_model = SpeakerRecognition.from_hparams(source="speechbrain/spkrec-ecapa-voxceleb",
savedir="pretrained_models/spkrec-ecapa-voxceleb")

# Speaker enrollment
path_enrollment = "enroll.wav"
signal, fs = torchaudio.load(path_enrollment)
embeddings_enroll = verif_model.encode_batch(signal)

# Test utterance
path_test = "test.wav"
signal, fs = torchaudio.load(path_test)
embeddings_test = verif_model.encode_batch(signal)

# Similarity
similarity = torch.nn.CosineSimilarity(dim=-1, eps=1e-6)
score = similarity(emb1, emb2)
threshold = 0.85
print(score>threshold)
```

References

- **Mexican NLP Summer School 2020**
- **Daniel Garica-Romero Presentation VoxSRC workshop 2020**
- **Book “Machine Learning for Speaker Recognition “ by Man-Wai Mak, Jen-Tzung Chien**





Thank you for your attention!!!