Deep learning based speaker recognition tutorial: Using ECAPA-TDNN as an example

National Univeristy of Singapore
Tao Ruijie
ruijie.tao@u.nus.edu

Overview: Speaker Recognition (SR)

1. Introduction: Definition, application, history and dataset.
2. How to **train** a SR model:
   - 2.1. Data format for training
   - 2.2. Feature extraction
   - 2.3. Data augmentation
   - 2.4. Speaker model
   - 2.5. Loss function & training performance
3. How to **evaluate** a SR model:
   - 3.1. Data format for evaluation
   - 3.2. Metrics for evaluation
   - 3.3. How to get the final scores
   - 3.4. Backend methods

About this tutorial

1. This project is **basic**, the target is to assist the researcher who just start this topic.

2. The whole tutorial will contain about 10 sessions, each session will has about 8-15 mins. Total time is about 90 mins.

3. Will try to introduce the code at the same time.

Annotation

NUS National University of Singapore

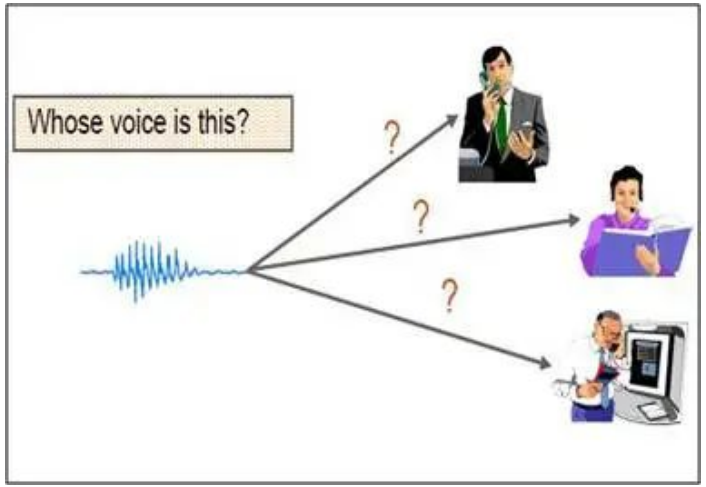
1. I am still an beginner of speaker recognition topic, this sildes&tutorial is only used to share some knowledge and coding experience of speaker recognition that I learnt.

2. If you find anything **wrong** in this tutorial, I apologize for that, please feel free to let me know. Thanks for your understanding!

3. Let’s build a better community!

1 Introduction: Definition

Speaker recognition (SR) is the identification of a person from characteristics of voices.

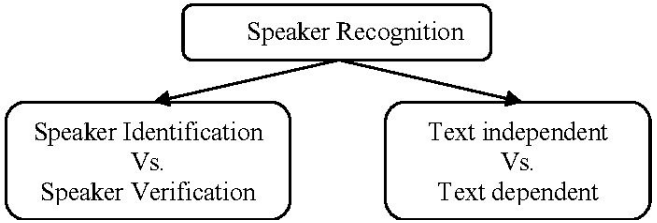


Face recognition

1 Introduction: Definition

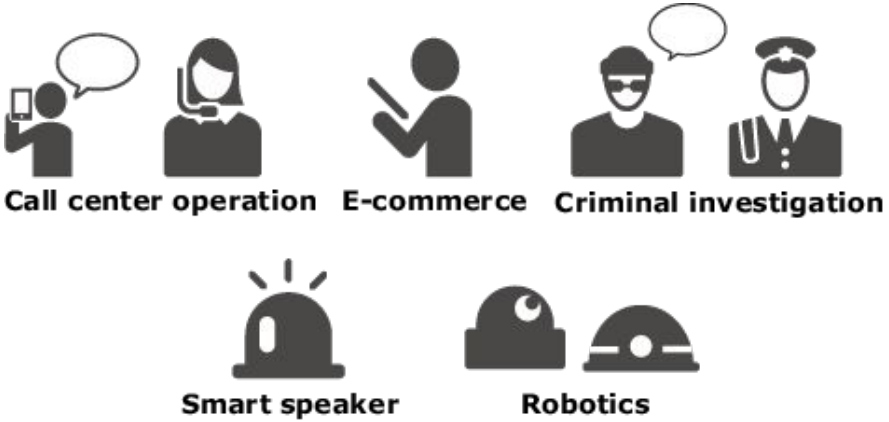
Speaker recognition (SR): One-to-many

Speaker verification (SV): One-to-one



1 Introduction: Application

1. Call center operation: Know the id of customer
2. E-commerce: Verify the user for payment
3. Criminal inverstigation: Similar to face recognition
4. Smart speaker and Robotics: Know who is talking
5. Security for phone and bank



1 Introduction: History

1987: Vector Quantization [1]

1997: Gaussian Mixture Model (**GMM**) [2]

2000: GMM-Universal Background Model (GMM-UBM) [3]

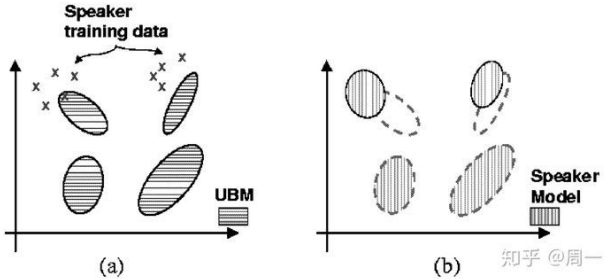
2006: Support vector machines (SVM) [4]

2007: Joint factor analysis (JFA) [5]

2010: **I-Vector** [6]

2018: **X-Vector** [7]

Others: ANNs, speaker-specific mapping, d-vector, end-to-end …



$$s = m + Vy + Ux + Dz$$

"Ideal" speaker supervector

Speaker-independent component

Speaker-dependent component

Channel-dependent component

Speaker-dependent residual component

$$s = m + Tw$$

Conversation side supervector

Total-variability matrix

i-vector

Ref: https://www.zhihu.com/people/leonjin

Layer	Layer context	Total context	Input x output
frame1	$[t-2, t+2]$	5	120x512
frame2	$\{t-2, t, t+2\}$	9	1536x512
frame3	$\{t-3, t, t+3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	1500Tx3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xN

[1] Text-dependent speaker verification using vector quantization source coding
[2] Robust text-independent speaker identification using Gaussian mixture speaker models
[3] Speaker verification using adapted gaussian mixture models
[4] Support vector machines for speaker and language recognition
[5] Joint factor analysis versus eigenchannels in speaker recognition
[6] Front-End Factor Analysis For Speaker Verification
[7] X-vectors robust dnn embeddings for speaker recognition

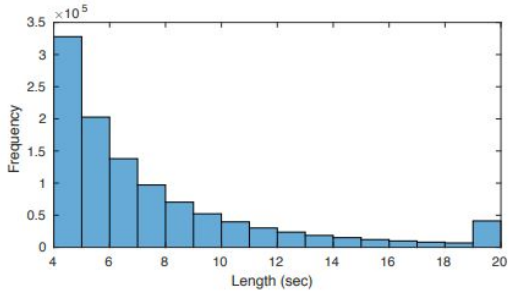
1 Introduction: Dataset

Other dataset for SR: NIST SRE(CTS SRE04-16, SRE18, SRE19…), Fisher, SITW, Switchbroad, Mixer 6, Dihard, CnCeleb, LRS2/3

Dataset we used in this project: VoxCeleb 1 & 2 [8][9]

	dev	test
# of speakers	1,211	40
# of videos	21,819	677
# of utterances	148,642	4,874

	dev	test
# of speakers	5,994	118
# of videos	145,569	4,911
# of utterances	1,092,009	36,237



Download data: https://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox2.html
Preprocess data: https://github.com/clovaai/voxceleb_trainer

[8] VoxCeleb: A large-scale speaker identification dataset
[9] VoxCeleb2: Deep speaker recognition

2.1 Data format for training & Dataloader

About the data for training

File Format: .wav

Length: Random 2 or 3 seconds from each utterance(fixed length)

(If duration is not enough, require padding)

Content: speech only, no VAD (voice activity detection)

2.1 Data format for training & Dataloader

Dataloader

“At the heart of PyTorch data loading utility is the torch.utils.data.DataLoader class. It represents a Python iterable over a dataset”

Create a custom Dataset class

```
class CustomTextDataset(Dataset):
    def __init__(self, txt, labels):
        self.labels = labels
        self.text = text

def __len__(self):
        return len(self.labels)

def __getitem__(self, idx):
        label = self.labels[idx]
        text = self.text[idx]
        sample = {"Text": text, "Class": label}
        return sample
```

How to iterate through the dataset when training a model

We will iterate through the Dataset without using ***collate_fn*** because its easier to see how the words and classes are being output by DataLoader. If the above function were used with ***collate_fn*** then the output would be tensors.

```
DL_DS = DataLoader(TD, batch_size=2, shuffle=True)

for (idx, batch) in enumerate(DL_DS):

    # Print the 'text' data of the batch
    print(idx, 'Text data: ', batch['Text'])

    # Print the 'class' data of batch
    print(idx, 'Class data: ', batch['Class'], '\n')
```

2.1 Data format for training & Dataloader

How to get the key, filename/path for training?

Read the official training list, map the speaker id to the class id.

```
id00012 id00012/21Uxsk56VDQ/00015.wav
id00012 id00012/2DLq_Kkc1r8/00016.wav
id00012 id00012/2DLq_Kkc1r8/00017.wav
id00012 id00012/2DLq_Kkc1r8/00018.wav
id00012 id00012/73OrGYvy4ng/00019.wav
id00012 id00012/C_FAL9gv8bo/00020.wav
id00012 id00012/C_FAL9gv8bo/00021.wav
id00012 id00012/C_FAL9gv8bo/00022.wav
id00012 id00012/C_FAL9gv8bo/00023.wav
id00012 id00012/C_FAL9gv8bo/00024.wav
id00012 id00012/C_FAL9gv8bo/00025.wav
id00012 id00012/C_FAL9gv8bo/00026.wav
id00012 id00012/C_FAL9gv8bo/00027.wav
```

File list: [‘id00012/C_FAL9gv8bo/00026.wav’,
‘id00012/C_FAL9gv8bo/00026.wav’,
‘id00015/HG2AS_DV241/00001.wav’,
…..]
Label list: [0,0,1,....]

2.2 Feature Extraction

Do we use the waveform directly for speaker recognition?

What is the feature extraction?

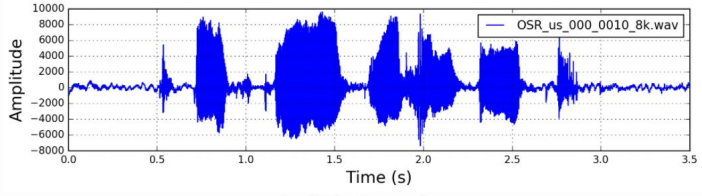
Why we need feature extraction?

How to do that?

If you feel this process is boring, you can only know one sentence: “In preprocess, we extract the frequency feature (which named Fbank) from the originial waveform to the speaker model.”

2.2 Feature Extraction: FBank

Original waveform: Sampling frequency (16k)



Signal in the Time Domain

Step 1: Pre-emphasis

Step 2: Frame blocking and windowing

Step 3: Fourier-Transform and Power Spectrum

Step 4: Filter Banks

Ref1: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html
Ref2: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
Ref3: https://zhuanlan.zhihu.com/p/276394091
Ref4: https://link.springer.com/content/pdf/bbm%3A978-3-319-49220-9%2F1.pdf

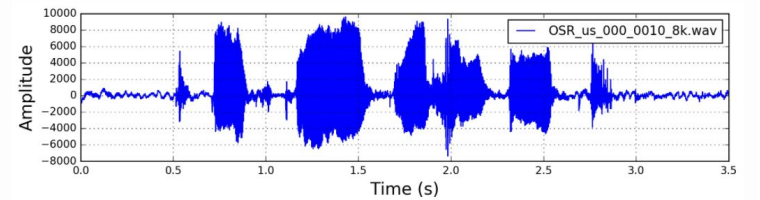
2.2 Feature Extraction: FBank

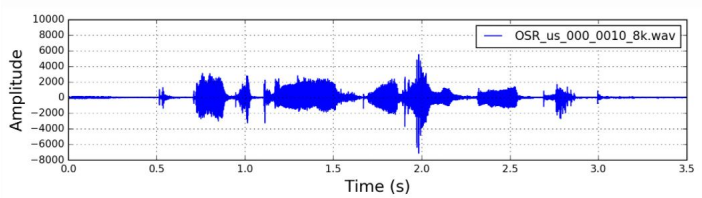
Step 1: Pre-emphasis

Amplify the high frequencies

$$y(t) = x(t) - \alpha x(t-1)$$

```
emphasized_signal = numpy.append(signal[0], signal[1:] - pre_emphasis * signal[:-1])
```

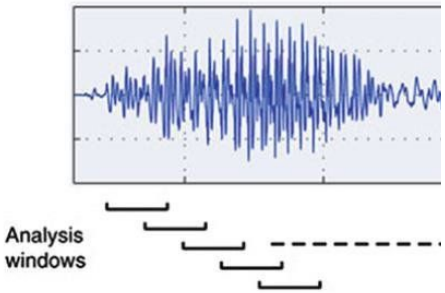


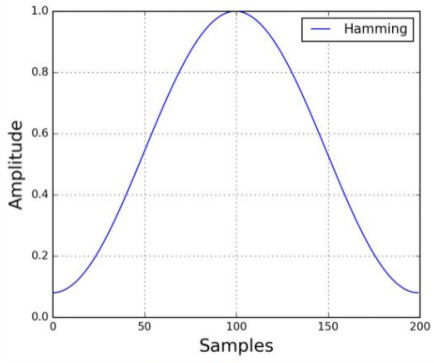


2.2 Feature Extraction: FBank

Step 2: Frame blocking and windowing

Method: split the signal into short-time frames





Hamming Window

2.2 Feature Extraction: FBank

Step 3: Fourier-Transform and Power Spectrum

Do an N-point FFT on each frame (Short-Time Fourier-Transform (STFT)) to calculate the **frequency spectrum**

Compute the power spectrum

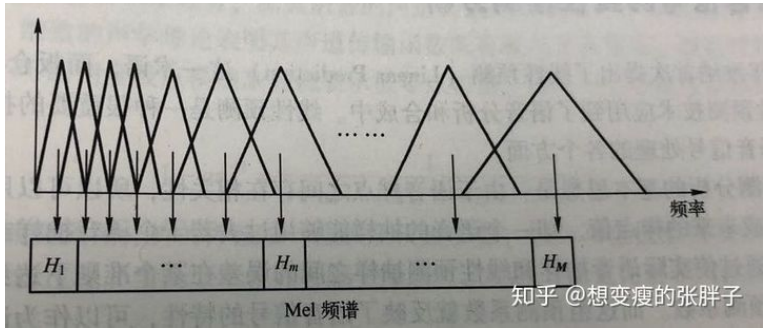
$$P = \frac{|FFT(x_i)|^2}{N}$$

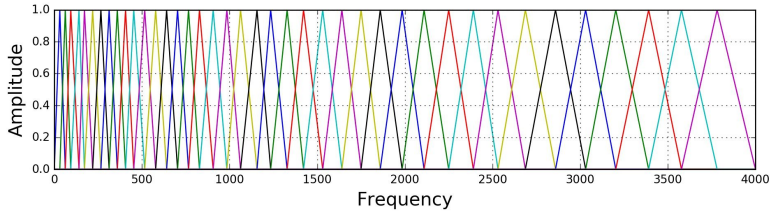
2.2 Feature Extraction: FBank

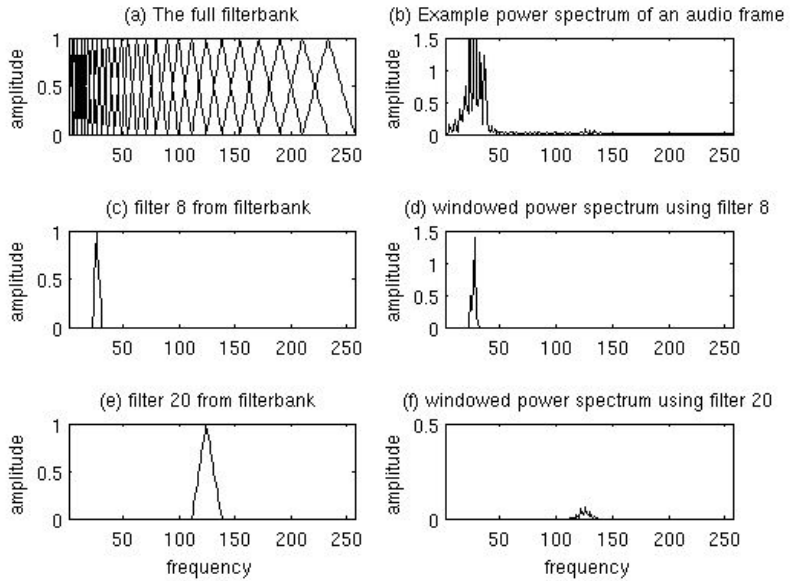
Step 4: Filter Banks (or Mel spectrum)

$$f_{Mel} = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

Mel spectrum is computed by passing the Fourier transformed signal through a set of band-pass filters known as Mel-filter bank







2.2 Feature Extraction: FBank

Step 5: Mel-frequency Cepstral Coefficients (MFCCs)

Discrete Cosine Transform (DCT)

Step 6: Mean normalization

Fbank: mean

MFCC: mean & var

2.2 Feature Extraction: Coding

Coding is very simple, there are many feature extraction toolkit. For instance:

librosa: https://librosa.org/doc/main/generated/librosa.feature.mfcc.html

python_speech_features: https://github.com/jameslyons/python_speech_features

pytorch: https://pytorch.org/audio/stable/_modules/torchaudio/transforms.html#MFCC

kaldi: https://kaldi-asr.org/doc/feat.html

Here I use pytorch as the example.

```
self.torchfbank = torch.nn.Sequential(
    PreEmphasis(),
    torchaudio.transforms.MelSpectrogram(sample_rate=16000, n_fft=512, win_length=400, hop_length=160, \
                                         f_min = 20, f_max = 7600, window_fn=torch.hamming_window, n_mels=80),
    )
```

2.3 Data Augmentation

What is data augmentation?

Make the speech noise & Hard to recognize

Why we need that?

This increases the amount and diversity of the existing training data, and achieves a significant improvement for the x-vector system.[7]

Is that useful?

Without that you can hardly get a very good result

[7] X-VECTORS: ROBUST DNN EMBEDDINGS FOR SPEAKER RECOGNITION

2.3 Data Augmentation

Dataset we need:

MUSAN [11]: There are 3 kinds of wav files (Speech&Noise&Music)

RIR [12]: Real world reverberation

[11] D. Snyder, G Chen, and D. Povey, “MUSAN: A Music, Speech, and Noise Corpus,”
[12] T. Ko, V. Peddinti, D. Povey, M. Seltzer, and S. Khudanpur, “A study on data augmentation of reverberant speech for robust speech recognition,”

2.3 Data Augmentation

Type of noise, Kaldi based:

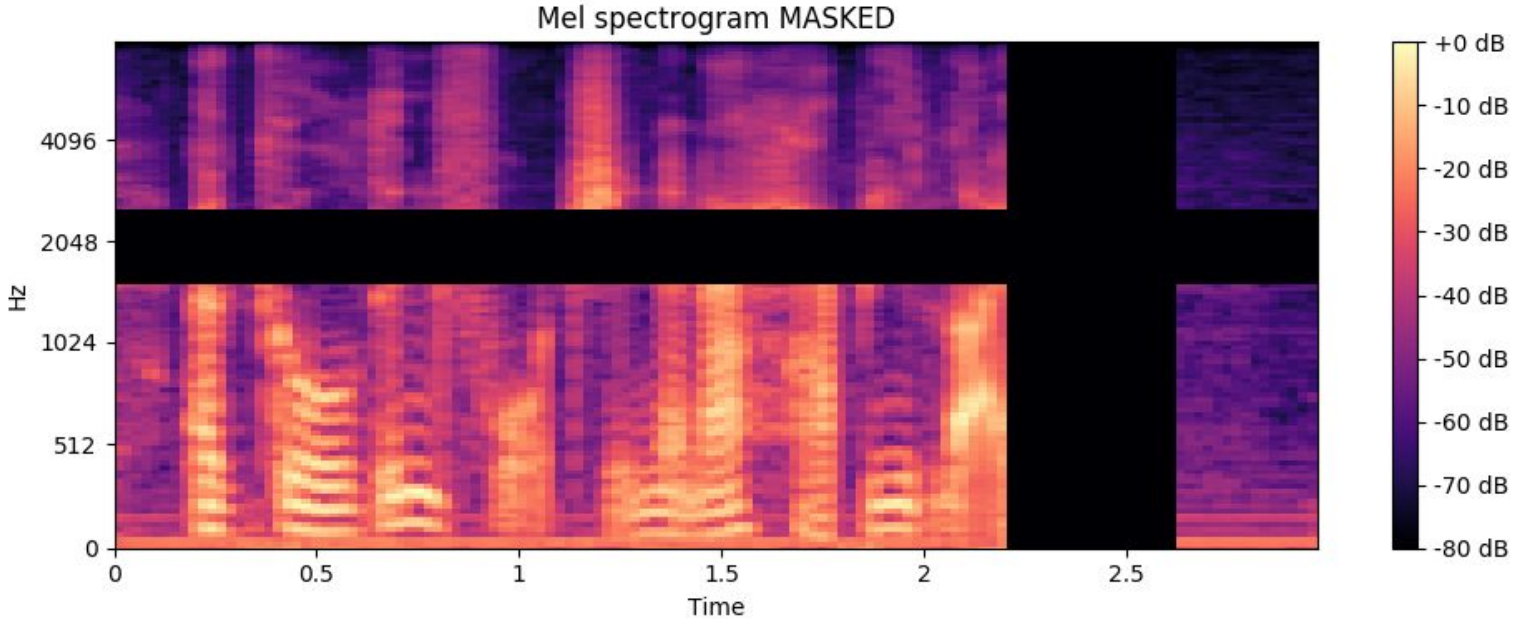
1. RIR
2. Music (add 1 music)
3. Noise (add 1 noise)
4. Babble (add 3-8 speech)

Others

1. Speaker Augmentation (change speed to get a new speaker)
2. Tempo up, tempo down, faster, slower
3. TV noise (add 1 speech + 1 music)
4. Spec Aug

2.3 Data Augmentation

Spec Aug



2.4 Speaker Model

NUS National University of Singapore

What next?

We use the speaker model to learn the speaker embedding.

Why we need that?

The input feature contains many information, we need the information specifically for speaker recognition (identity information).

What is the speaker embedding?

We hope the speaker embedding contains the identity related speaker information, this embedding is very smaller than the orginal speech feature.

2.4 Speaker Model

How to achieve that?

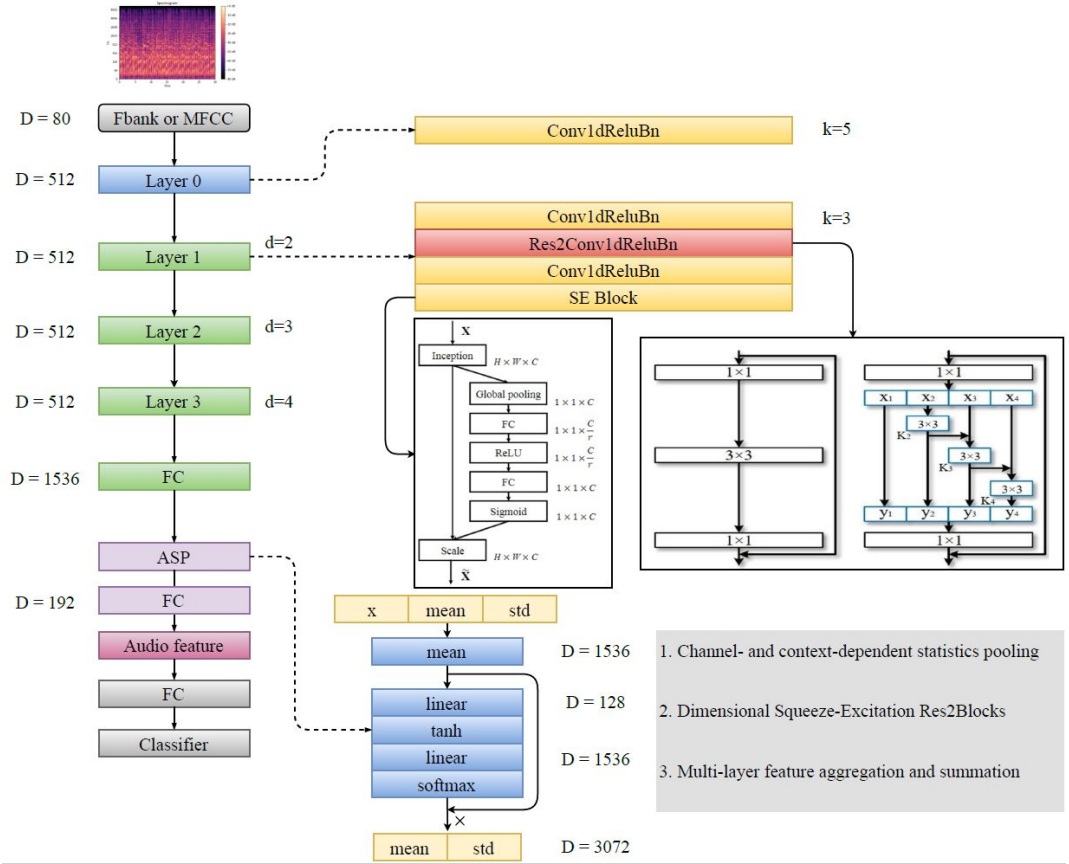
We use a speaker model (neural network) to learn a mapping or extractor, which from the input speech features, to the speaker embedding.

During training, we have the speaker label for each utterance, so a supervised method can be used to guide the model learn that mapping.

Here we use ECAPA-TDNN as the example [13]

[13] B. Desplanques, J. Thienpondt, and K. Demuynck, “ECAPA- TDNN: Emphasized channel attention, propagation and aggrega- tion in TDNN based speaker verification,

2.4 Speaker Model: ECAPA-TDNN Model



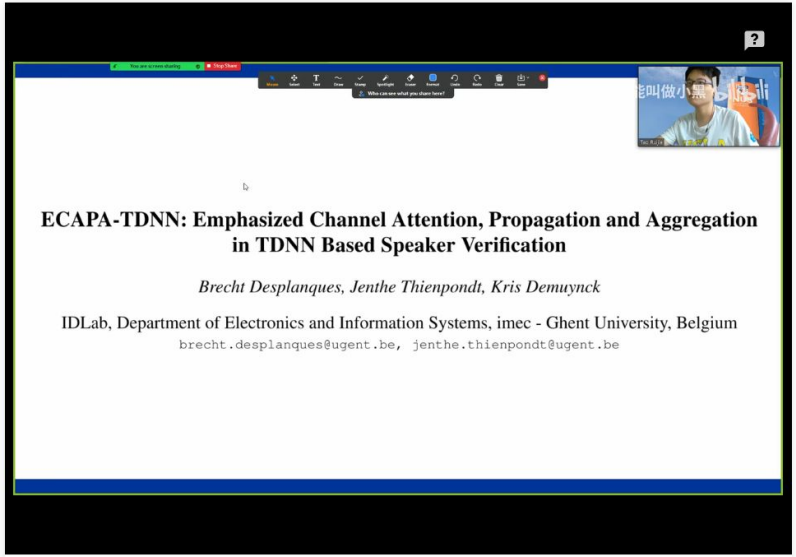
2.4 Speaker Model: ECAPA-TDNN Model

The details for the model and the code can be found in my previous video:

https://www.bilibili.com/video/BV1fQ4y1Y75w/

速通-声纹识别-强力模型-ECAPA-TDNN。半小时，图文并茂，层层剖析ECAP…

853播放 · 总弹幕数1　2021-08-25 16:16:23



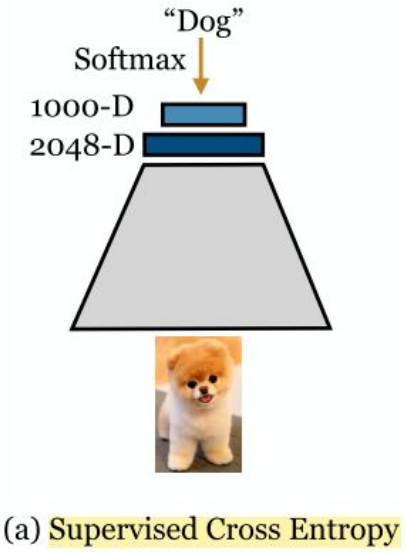
Here I describe more for the code in

https://github.com/TaoRuijie/ECAPATDNN

2.5 Loss function

What is the loss function?

Prediction vs Ground Truth



(a) Supervised Cross Entropy

What is the loss function in speaker recognition?

Classification loss

How to connect the speaker embedding to the classification loss?

Classifier or classification layer

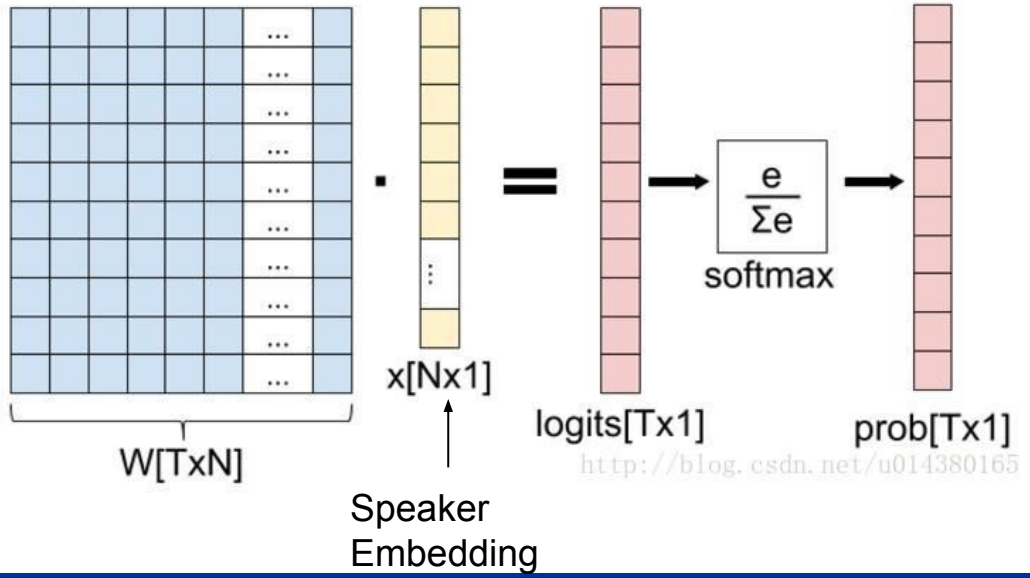
2.5 softmax with cross-entropy loss

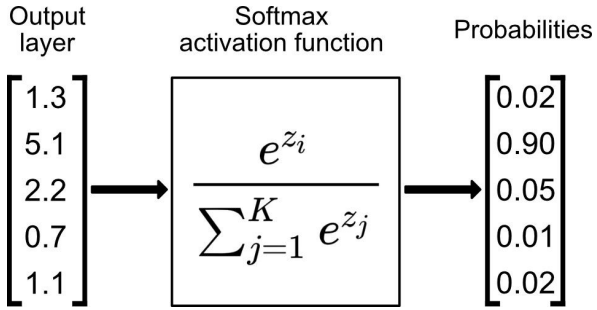
Softmax:

N = 192 (Dimension of speaker embedding)

T = 5994 (Number of speaker & class)

Ground Truth: [T * 1] (One hot)

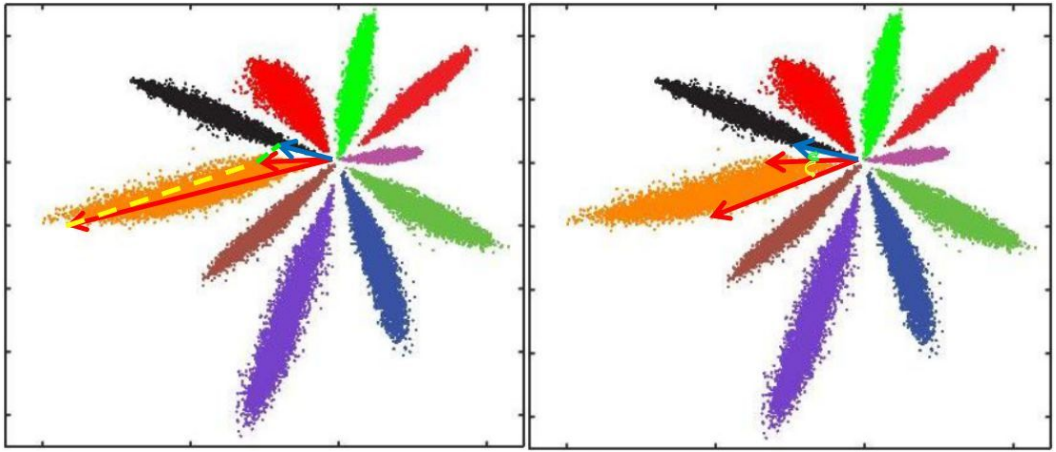




$$L(\hat{y}, y) = -\sum_{k}^{K} y^{(k)} \log \hat{y}^{(k)}$$

2.5 Problem of softmax

L2 distance and cos distance

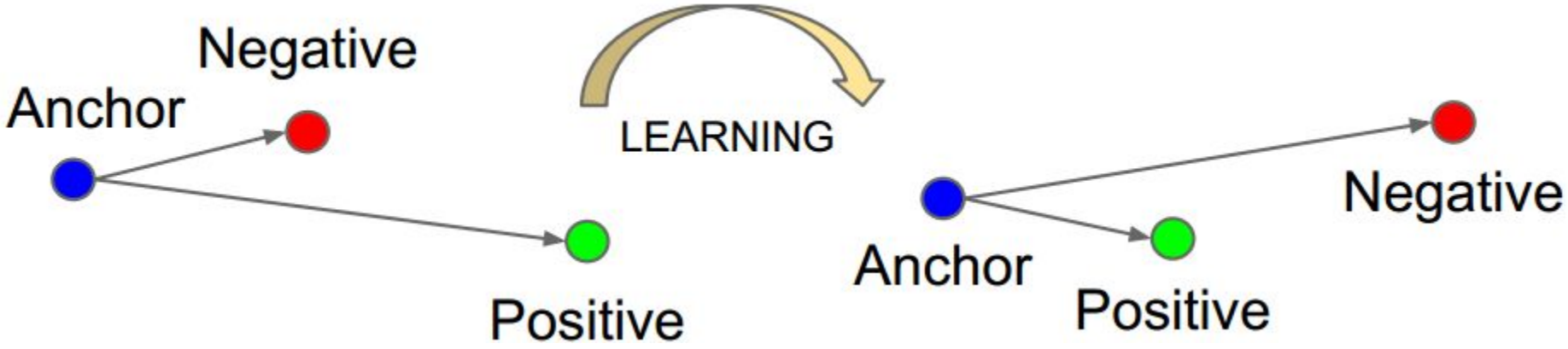


Specific for face recognition & speaker recognition:

1. Data in the same class: distance become smaller
2. Data in the different class: distance become larger

Based on that, there are many well-performed loss function

2.5 Triple Loss [14]



$$\sum_{i}^{N}\left[\|f(x_i^a)-f(x_i^p)\|_2^2-\|f(x_i^a)-f(x_i^n)\|_2^2+\alpha\right]_+$$

[14] Facenet: A unified embedding for face recognition

https://zhuanlan.zhihu.com/p/34404607

2.5 L-Softmax [15]

Make the angle for the target class smaller

$$L_i = -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(\theta_{y_i})}}{\sum_j e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)}}\right)$$ Softmax Loss

$$L_i = -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})}}{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)}}\right)$$

$$\psi(\theta) = (-1)^k\cos(m\theta) - 2k, \quad \theta \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$$ L-Softmax Loss

[15v] Large-Margin Softmax Loss for Convolutional Neural Networks

2.5 SphereFace [16]

Make the angle for the target class smaller

$$L_{\text{ang}} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\boldsymbol{x}_i\|\psi(\theta_{y_i,i})}}{e^{\|\boldsymbol{x}_i\|\psi(\theta_{y_i,i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\right)$$

$$\psi(\theta) = (-1)^k \cos(m\theta) - 2k, \quad \theta \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$$

Add the weight norm, also call A-softmax

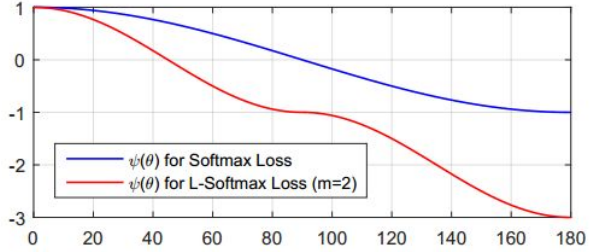


Figure 3. $\psi(\theta)$ for softmax loss and L-Softmax loss.

[16] SphereFace: Deep Hypersphere Embedding for Face Recognition

2.5 Center Loss [17]

Make the angle for the target class smaller

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_C$$

$$= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \| \boldsymbol{x}_i - \boldsymbol{c}_{y_i} \|_2^2$$

[17] A discriminative feature learning approach for deep face recognition

2.5 Feature Normalization

Feature norm & Scale factor

$$\hat{\boldsymbol{c}}_k = \frac{\boldsymbol{c}_k}{\|\boldsymbol{c}_k\|}, \quad \hat{\boldsymbol{f}}^{(i)} = \frac{\alpha \boldsymbol{f}^{(i)}}{\|\boldsymbol{f}^{(i)}\|}, \quad p_k^{(i)} = \frac{\exp(\hat{\boldsymbol{c}}_k^T \cdot \hat{\boldsymbol{f}}^{(i)})}{\sum_m \exp(\hat{\boldsymbol{c}}_m^T \cdot \hat{\boldsymbol{f}}^{(i)})}$$

Why feature norm:

1. **Make the system focus on the angle**
2. **L2 dis = cos dis**

Why scale factor:

1. **Make a bigger ball space with more information**
2. **Deal with the feature norm's training problem**

2.5 AMsoftmax & Cosface [18][19]

cos(mθ) to cos(θ) - m

$$\mathcal{L}_{AMS} = -\frac{1}{n}\sum_{i=1}^{n} log\frac{e^{s\cdot(cos\theta_{y_i}-m)}}{e^{s\cdot(cos\theta_{y_i}-m)}+\sum_{j=1,j\neq y_i}^{c} e^{s\cdot cos\theta_j}}$$

$$L_{lmc} = \frac{1}{N}\sum_{i} -\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)}+\sum_{j\neq y_i} e^{s\cos(\theta_{j,i})}}, \tag{4}$$

subject to

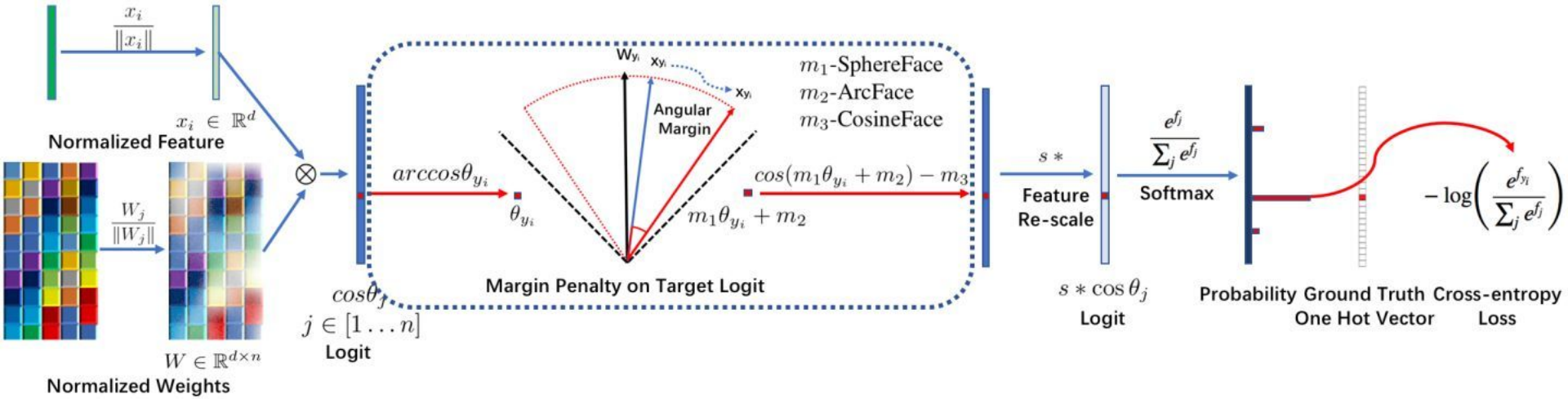
$$\begin{aligned} W &= \frac{W^*}{\|W^*\|}, \\ x &= \frac{x^*}{\|x^*\|}, \\ cos(\theta_j, i) &= {W_j}^T x_i, \end{aligned} \tag{5}$$

[18] Additive Margin Softmax for Face Verification
[19] CosFace: Large Margin Cosine Loss for Deep Face Recognition

2.5 AAM softmax [20]

cos(θ) - m to cos(θ+m)

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_j}}$$



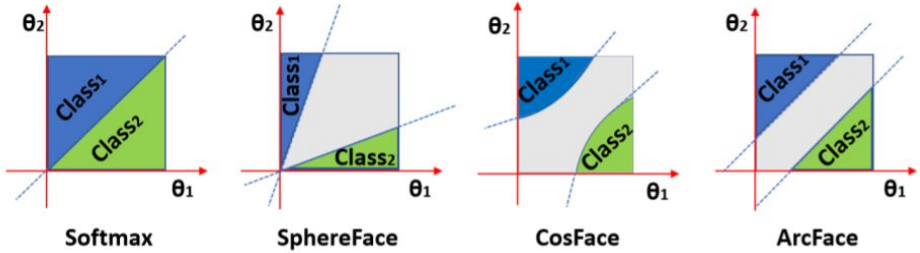
[20] ArcFace: Additive Angular Margin Loss for Deep Face Recognition

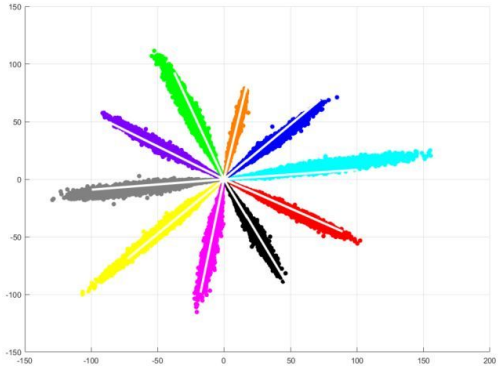
2.5 Why large margin

Compared with cos(θ) [softmax],
cos(θm) [Sphereface], cos(θ) - m [AMsoftmax], cos(θ + m) [AAMsoftmax]

1. They are more difficult to meet the training requirement.

2. Lead to a smaller θ for each sample in the class





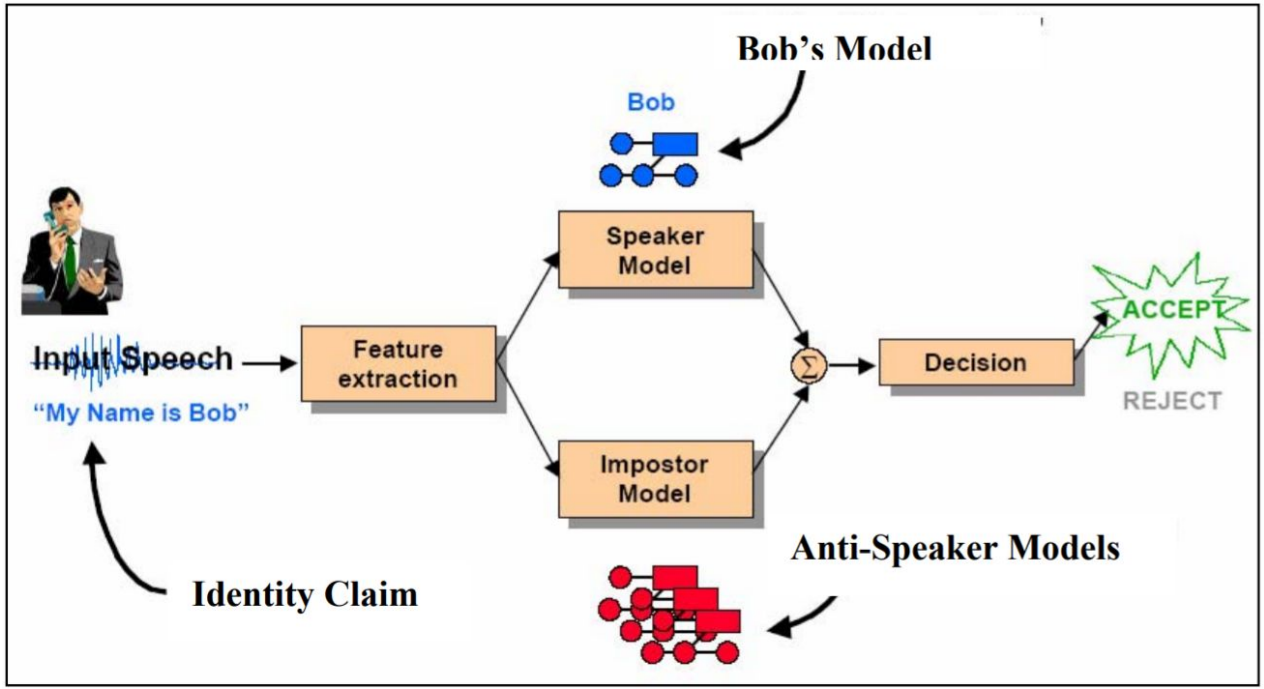
[20] ArcFace: Additive Angular Margin Loss for Deep Face Recognition

What next?

NUS National University of Singapore

Talk is cheap,
show me your performance.

3.1 Data format for evaluation



3.1 Evaluation file format

```
1 id10270/x6uYqmx31kE/00001.wav id10270/8jEAjG6SegY/00008.wav
0 id10270/x6uYqmx31kE/00001.wav id10300/ize_eiCFEg0/00003.wav
1 id10270/x6uYqmx31kE/00001.wav id10270/GWXuj1-xAVM/00017.wav
0 id10270/x6uYqmx31kE/00001.wav id10273/0OCW1HUxZyg/00001.wav
1 id10270/x6uYqmx31kE/00001.wav id10270/8jEAjG6SegY/00022.wav
0 id10270/x6uYqmx31kE/00001.wav id10284/Uzxv7Axh3Z8/00001.wav
1 id10270/x6uYqmx31kE/00001.wav id10270/GWXuj1-xAVM/00033.wav
0 id10270/x6uYqmx31kE/00001.wav id10284/7yx9A0yzLYk/00029.wav
1 id10270/x6uYqmx31kE/00002.wav id10270/5r0dWxy17C8/00026.wav
0 id10270/x6uYqmx31kE/00002.wav id10285/m-uILToQ9ss/00009.wav
1 id10270/x6uYqmx31kE/00002.wav id10270/GWXuj1-xAVM/00035.wav
0 id10270/x6uYqmx31kE/00002.wav id10306/uzt36PBzT2w/00001.wav
1 id10270/x6uYqmx31kE/00002.wav id10270/GWXuj1-xAVM/00038.wav
0 id10270/x6uYqmx31kE/00002.wav id10307/kp_GCjLq4qA/00004.wav
1 id10270/x6uYqmx31kE/00002.wav id10270/GWXuj1-xAVM/00033.wav
```

1 means postive pair
0 means negative pair

These utterances come from the unkown speakers !

3.1 Pipeline for evaluation

Input: Utterance A and utterance B
(Raw wav files, without data augmentation)

Then: Get speaker embedding from A and B
(Do not need the classification layer now!)

Output: The similarity score between these two utterances

3.2 Metrics for evaluation

trials.txt

```
1 id10912/H5qe-mHhOyQ/00007.wav id10912/nDpaVYtKQUo/00004.wav
1 id10560/p_V0oeCcc0w/00011.wav id10560/_SIZKabFLAM/00001.wav
0 id10792/La6IDPsWJHE/00007.wav id11044/BVzEyYdVLXg/00001.wav
1 id10294/Jr2KDTTWYgM/00001.wav id10294/ba0ZhyVTodw/00006.wav
0 id10322/wf56LXLXPks/00010.wav id10413/1wqsLG0T27I/00003.wav
1 id10114/5G2weYmtz88/00006.wav id10114/CkBmw0zFP0I/00010.wav
1 id11212/Rp83Xx3bSjE/00033.wav id11212/ygFqhQop1Yc/00013.wav
0 id11065/10JyP3MFPjc/00026.wav id10587/31ZR8xeMUCA/00004.wav
1 id10009/sQIqfA-I_Ew/00001.wav id10009/AtavJVP4bCk/00005.wav
0 id11014/pYDoUI3vHzY/00002.wav id10258/z7PnkmwoByo/00018.wav
```

Performance?

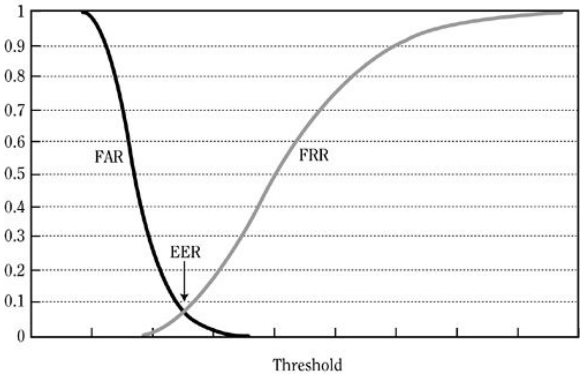
scores.txt

```
0.503 id10912/H5qe-mHhOyQ/00007.wav id10912/nDpaVYtKQUo/00004.wav
0.431 id10560/p_V0oeCcc0w/00011.wav id10560/_SIZKabFLAM/00001.wav
0.351 id10792/La6IDPsWJHE/00007.wav id11044/BVzEyYdVLXg/00001.wav
0.586 id10294/Jr2KDTTWYgM/00001.wav id10294/ba0ZhyVTodw/00006.wav
0.338 id10322/wf56LXLXPks/00010.wav id10413/1wqsLG0T27I/00003.wav
0.564 id10114/5G2weYmtz88/00006.wav id10114/CkBmw0zFP0I/00010.wav
0.466 id11212/Rp83Xx3bSjE/00033.wav id11212/ygFqhQop1Yc/00013.wav
0.360 id11065/10JyP3MFPjc/00026.wav id10587/31ZR8xeMUCA/00004.wav
```

Equal Error Rate (EER) and Minimum Detection Cost (MinDCF)

3.2 Metrics for evaluation: EER

Label \ Prediction	0	1
0	TN	FP
1	FN	TP



T or F: Based on the “comparison”
P or N: Based on the “prediction” itself

False Acceptance Rate (FAR), label is 0, prediction is 1.

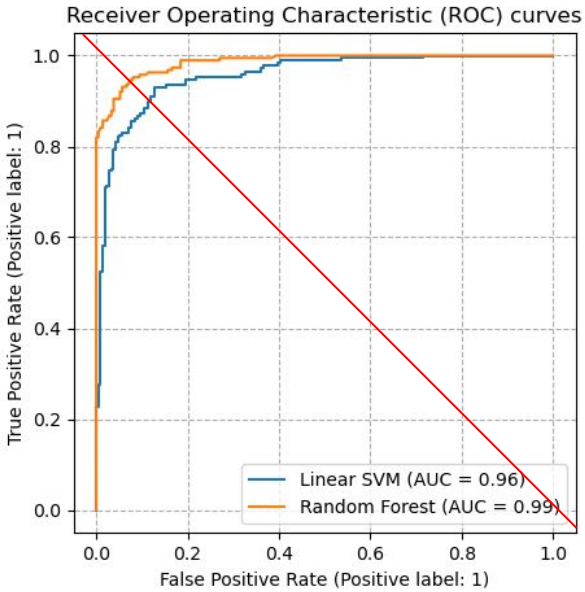
$$FAR = FP/(TN + FP)$$

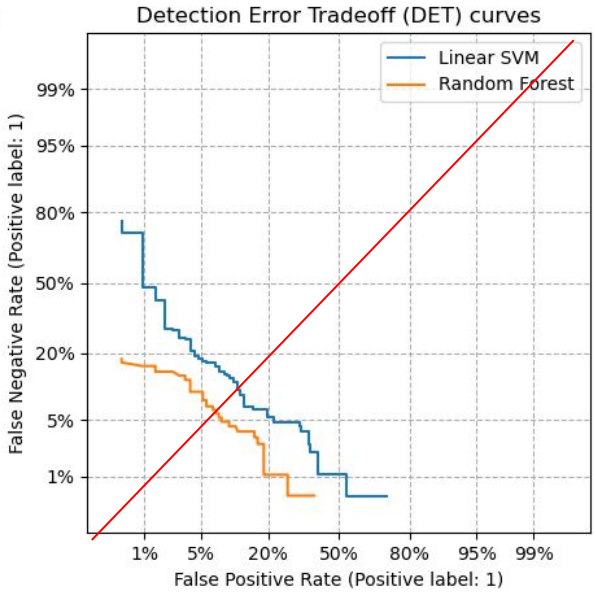
False Rejection Rate (FRR), label is 1, prediction is 0.

$$FRR = FN/(FN + TP)$$

3.2 Metrics for evaluation: EER

$$FAR = FP/(TN + FP) \quad FRR = FN/(FN + TP) \quad EER = FAR = FRR$$





1) Lower is better. 2) For EER, FAR and FRR is equally important

3.2 Metrics for evaluation: minDCF

$$C_{Det}(\theta) = C_{Miss} \times P_{Target} \times P_{Miss}(\theta) + C_{FalseAlarm} \times (1 - P_{Target}) \times P_{FalseAlarm}(\theta)$$

(NIST SRE18 Plan)

Miss: False Reject
FalseAlarm: False Accept

Setting in VoxCeleb2

1. CMiss (cost of a missed detection) = 1
2. CFalseAlarm (cost of a spurious detection) = 1
3. PTarget (a priori probability of the specified target speaker) = 0.05

1) Lower is better. 2) In this setting, FAR is more important than FRR

3.3 How to get the final scores

Step 1: Read all the test utterance's name from the given list
Step 2: Extract the speaker embedding for each utteranace, Norm
Step 3: For each Pair, utterance A and B, compute cosine similarity to get the final score.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

$$L_2 norm\ d(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)}$$

$$\begin{aligned} ||\mathbf{x} - \mathbf{y}||_2^2 &= (\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y}) \\ &= \mathbf{x}^\top \mathbf{x} - 2\mathbf{x}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y} \\ &= 2 - 2\mathbf{x}^\top \mathbf{y} \\ &= 2 - 2\cos\angle(\mathbf{x}, \mathbf{y}) \end{aligned}$$

3.3 How to get the final scores

L2 euclidean distance vs cosine distance

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

$$L_2 norm\ d(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

$$\begin{aligned} ||\mathbf{x} - \mathbf{y}||_2^2 &= (\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y}) \\ &= \mathbf{x}^\top \mathbf{x} - 2\mathbf{x}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y} \\ &= 2 - 2\mathbf{x}^\top \mathbf{y} \\ &= 2 - 2\cos\angle(\mathbf{x}, \mathbf{y}) \end{aligned}$$

Conclusion: that will not effect the performance if you use norm!

3.3 Trick for the score

Trick: Two kinds of scores to improve the performance

For utterance A and B:

Score 1: extract 1 embedding for the entire A or B, compute score

Score 2: extract 5 embedding from 5 equally-spaced 3-seconds segments in A and B, compute the score matrix and get the average

Final score = score 1 + score 2, can get about 5% improvement

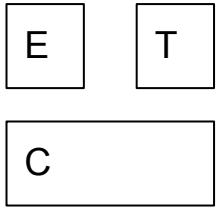
3.4 Backend methods

Score Norm, a useful method to improve the performance. [21]

The goal of score normalization is to reduce within trial variability leading to improved performance, better calibration, and more reliable threshold setting

[21] Analysis of Score Normalization in Multilingual Speaker Recognition

3.4 Backend methods: Z-norm



2.1. Z-norm

Zero score normalization [4] employs impostor score distribution for enrollment file. It uses a cohort $\mathcal{E} = \{\varepsilon_i\}_{i=1}^{N}$ with N speakers which we assume to be different from the speakers in utterances e and t. The cohort scores are

$$S_e = \{s(e, \varepsilon_i)\}_{i=1}^{N} \tag{1}$$

and are formed by scoring enrollment utterance e with all files from cohort $\mathcal{E}$. The normalized score is then:

$$s(e,t)_{z\text{-}norm} = \frac{s(e,t) - \mu(S_e)}{\sigma(S_e)}, \tag{2}$$

where $\mu(S_e)$ and $\sigma(S_e)$ are mean and standard deviation of S_e.

3.4 Backend methods: T-norm

2.2. T-norm

Test score normalization [7] is similar to Z-norm with the difference that it normalizes the impostor score distribution for the test utterance. T-norm can be expressed by:

$$S_t = \{s(t, \varepsilon_i)\}_{i=1}^{N} \tag{3}$$

$$s(e,t)_{t\text{-}norm} = \frac{s(e,t) - \mu(S_t)}{\sigma(S_t)} \tag{4}$$

where $\mu(S_t)$ and $\sigma(S_t)$ are mean and standard deviation of S_t.

3.4 Backend methods: ZT-norm

2.3. ZT-norm

ZT-norm or TZ-norm use Z- and T-norm in series, and might use different cohorts for each step [22]. By doing this, the scores are normalized with respect to both enrollment and test utterances. Applying ZT-norm for Joint Factor Analysis (JFA) based system was an essential step, which improved results by 50% relative [5, 6].

3.4 Backend methods: S-normc

2.4. S-norm

The symmetric normalization (S-norm) computes an average of normalized scores from Z-norm and T-norm [11]. S-norm is symmetrical as $s(e,t) = s(t,e)$, while the previously mentioned normalizations depend on the order of e and t.

$$s(e,t)_{s\text{-}norm} = \frac{1}{2} \cdot (s(e,t)_{z\text{-}norm} + s(e,t)_{t\text{-}norm})$$

$$= \frac{1}{2} \cdot \left(\frac{s(e,t) - \mu(S_e)}{\sigma(S_e)} + \frac{s(e,t) - \mu(S_t)}{\sigma(S_t)} \right)$$

3.4 Backend methods: Adaptive S-norm

In adaptive T-norm [17] or Top-norm [18], only part of the cohort is selected[2] to compute mean and variance for normalization. We investigated the same cohort selection for Z-norm, T-norm, ZT-norm and S-norm - we call this selection adaptive, as the selected cohort might change for every speaker.

Two variants of adaptive cohort selection can be found in the literature: the adaptive cohort can be either selected to be X closest (most positive scores) files to the enrollment file $\mathcal{E}_e^{top}$, or, as in [20], to the test file $\mathcal{E}_t^{top}$. We have to note that such cohorts are different for each enrollment utterance e or test utterance t respectively. The cohort scores based on such selections for the enrollment utterance are then:

$$S_e(\mathcal{E}_e^{top}) = \{s(e,\varepsilon)\}_{\forall \varepsilon \in \mathcal{E}_e^{top}}, \quad S_e(\mathcal{E}_t^{top}) = \{s(e,\varepsilon)\}_{\forall \varepsilon \in \mathcal{E}_t^{top}} \tag{6}$$

and correspondingly for the test utterance t.

Two variants were investigated with S-norm: the normalized score for the first one called **adaptive S-norm1** is

$$s(e,t)_{as\text{-}norm1} = \frac{1}{2} \cdot \left(\frac{s(e,t) - \mu(S_e(\mathcal{E}_e^{top}))}{\sigma(S_e(\mathcal{E}_e^{top}))} + \frac{s(e,t) - \mu(S_t(\mathcal{E}_t^{top}))}{\sigma(S_t(\mathcal{E}_t^{top}))} \right) \tag{7}$$

and the second variant, **adaptive S-norm2**, is defined as

$$s(e,t)_{as\text{-}norm2} = \frac{1}{2} \cdot \left(\frac{s(e,t) - \mu(S_e(\mathcal{E}_t^{top}))}{\sigma(S_e(\mathcal{E}_t^{top}))} + \frac{s(e,t) - \mu(S_t(\mathcal{E}_e^{top}))}{\sigma(S_t(\mathcal{E}_e^{top}))} \right) \tag{8}$$

From my experience, Adaptive S-norm can improve the result by about 10%

End

Try to enjoy your research !

Try to do meaningful research forever !

Life is more important than study !