

1 이론적 정리

이 섹션에서는 그동안 연구실에서 화자 인식(Speaker Recognition)에 대하여 공부하면서 알게 된 사실들을 정리한다. 초반부에서는 화자 인식 기술이 어떻게 발전하고 있는지를 다루고, 그 다음부터는 DNN을 이용한 화자인식(DSR) 기법을 소개한다.

1.1 화자 인식의 정의

화자 인식(Speaker Recognition)은 어떤 화자의 발성이 있을 때, 그 발성을 가공하여 화자에 대한 정보를 추측하는 것을 말한다. 이러한 화자 인식 기술은 어떤 문제를 해결하고자 하는지에 따라 크게 **화자 식별(Speaker Identification)**, **화자 인증(Speaker Verification)**, **화자 구분(Speaker Diarization)**으로 나뉘는데, 각각에 대한 설명은 아래와 같다.[10]

- **화자 식별(Speaker Identification)**

어떤 발성 데이터를 토대로 그 발성이 사전에 등록된 복수의 화자들 중 누구와 가장 가까운지를 판별하는 태스크이다. 일반적으로 여러 화자들의 발성 데이터를 사전에 준비한 뒤 화자 식별을 위한 모델에 학습시키는 과정이 선행되는데, 이를 **등록(Enrollment)** 단계라고 부른다. 화자들의 발성 데이터를 등록하였다면 그 이후에는 모델에 임의의 발성 데이터를 입력하는 것으로 그 발성이 누구의 발성과 가장 유사한지를 판별할 수 있다.

- **화자 인증(Speaker Verification)**

어떤 발성 데이터가 사전에 등록된 한 사람의 발성이 맞는지를 검증하는 태스크이다. 즉, 앞에서 살펴본 화자 식별 태스크는 사전에 여러 화자들이 등록되고 나중에 입력된 발성이 누구의 발성인지 를 알아내는 것이 목적이지만, 화자 인증은 단 한 사람의 발성만 등록한 뒤, 나중에 들어오는 발성이 그 화자의 것이 맞는지를 검증한다.

- **화자 구분(Speaker Diarization)**

화자 구분 태스크는 화자 식별과 유사하지만 조금 더 일반적인 태스크로, 화자 식별의 경우 발성 데이터에 한 사람만 존재한다는 것이 보장되지만, 화자 구분 태스크에서는 한 발성 데이터에 여러 사람이 동시에 말하는 경우가 있을 수 있고, 시간별로 어떤 사람들이 말을 하고 있는지를 구분해내는 태스크이다. 보통 음성 데이터에서 자동으로 화자들 간 대화 정보를 수집하기 위한 시스템을 구현할 때 많이 이용되고, 등록 절차를 밟지 않고 곧바로 화자를 구분해낼 수 있다.

그런데 화자 식별을 위한 모델을 구현할 때 유의해야 할 점은, 만약 등록된 적이 없는 사람의 발성을 입력으로 주었다면 그 사람의 발성 데이터는 모델에 등록된 적이 없음에도 불구하고 유사도가 가장 높은 사람을 발성의 주인이라고 예측한다는 점이다. 이러한 문제점은 화자

식별 결과 도출한 화자가 실제 그 발성을 한 사람인지 를 검증하는 화자 인증 태스크를 추가로 시행함으로서 해결할 수 있다.

위에서는 화자 인식 기술을 목적에 따라 나누었다면, 사용하는 발성 데이터가 정해진 구절을 말하는 데이터인지, 그렇지 않고 임의의 구절을 말하는 데이터인지에 따라 문장 종속(Text-Dependent)과 문장 독립(Text-Independent)로 나눌 수 있다. 각각에 대한 설명은 아래와 같다.[10]

- **문장 종속(Text-Dependent)**

사전에 정해진 문장만 발음하는 발성만 모델에 등록 및 테스트하는데 사용되는 경우를 지칭한다. 항상 같은 문장만을 발음한다는 점에서 화자 인식의 정확도가 문장 독립에 비해서 높지만, 항상 같은 문장만을 발음해야 모델을 이용할 수 있으므로 사용자 입장에서는 불편한 경우이다.

- **문장 독립(Text-Independent)**

발성 데이터에서 발음하는 문장이 정해져 있지 않고 임의의 문장을 발음하는 발성도 허용하는 경우를 지칭한다. 발음하는 문장이 정해져 있지 않기 때문에 문장 종속인 경우보다 화자 인식의 난이도가 높고 이 때문에 모델의 성능도 문장 종속보다 낮은 편이다. 하지만 사용자 입장에서는 어떤 문장을 말해도 모델을 이용할 수 있으므로 편리성은 높다.

1.2 DNN을 이용한 화자 인식

초창기의 화자 인식 시스템은 Gaussian mixture models(GMM), Hidden Markov models(HMM), universal background models(UBM)을 통해 이루어졌고, 이후 GMM과 UBM을 합친 GMM-UBM이 등장하면서 i-vector system이 제안되며 되었다. i-vector 시스템이 등장하면서 본격적으로 화자 인식의 신뢰성이 상승하게 되었고, i-vector는 오늘날에도 여러 플랫폼에서 baseline으로 이용되고 있다.[2]

GMM이나 i-vector 이후에는 더 이상 handcraft statistical means에 의존하지 않고 deep learning을 이용하여 시스템을 구현하는 연구가 활발히 진행되고 있다. 화자 인식 태스크에 DNN을 적용하게 되면서, 예측의 정확도가 크게 상승하였고 robust한 모델을 훈련시킬 수 있었다.

DNN을 이용하여 화자 인식 태스크를 수행하는 것을 특별히 **Deep Speaker Recognition(DSR)**이라고 부른다. 그리고 구현되는 방식에 따라 음성 데이터를 별도의 가공 없이(raw-waveform인 상태로) DSR을 수행하는 방식과 음성 데이터에 별도의 전처리를 거친 뒤 DSR을 수행하는 방식으로 나뉜다.

- **Raw-waveform으로 DSR** 음성 데이터에 별도의 가공을 하지 않고 waveform의 양자화 된 값에 따라 normalization만 가한 뒤 바로 DNN에 입력하는 방식이다. 이러한 방식은 음성 데이터의 time domain과 frequency domain 간의 관계를 적극적으로 이용하기 어렵기 때문에 이 방식으로 제안된 시스템은 상대적으로 적다.[2] 하지만 RawNet과 같이 raw-waveform을 이용했음에도 좋은 성능을 보인 사례가 존재한다.
- **전처리를 거치는 DSR** 음성 데이터를 그대로 사용하지 않고 음성 데이터를 일정 시간 간격으로 분할한 뒤 분할된 각 utterance에 전처리를 하여 DSR을 수행하는 방식이다. 음성 데이터가 연속된 긴 시간 동안 수집된 신호값으로 표현되기 때문에 이를 한번에 모두 사용하지 않고, 일정한 시간 간격으로 분할하여 utterance 단위로 사용하는 경우가 많다. 이렇게 만든 몇 초 정도의 utterance는 한 단어 정도의 음성을 담고 있는데, 이를 다시 20~500 milliseconds 정도의 frame 단위로 나누고 전처리(pre-processing)를 하여 모델의 입력으로 사용한다. 오늘날 주로 사용하는 전처리 방식은 크게 3가지가 존재하는데, **Spectrogram**, **Mel-filterbank**, **MFCC**가 있다. Figure 1이 3가지 전처리 방식의 진행 과정을 나타내는데, 해당 그림을 통해서 알 수 있듯, Spectrogram, Filterbank, MFCC 순으로 데이터에 가하는 처리가 적음을 알 수 있다. 또한 전처리에는 trainable weights가 없으므로 모델의 학습 과정과 독립적으로 진행된다.

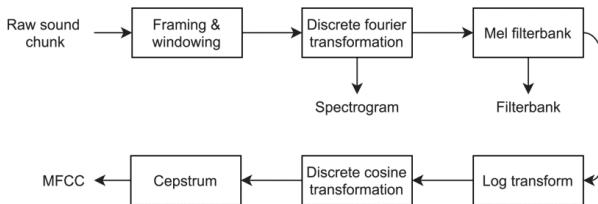


Figure 1: 전처리(pre-processing)의 과정을 나타내는 figure.

DNN을 화자 인식에 사용하는 DSR을 위해 지금까지 매우 다양한 Architecture가 제안되었는데, 이는 시스템의 구성에 따라 크게 **Stage-wise DSR**과 **End-to-end DSR**로 나뉜다.

1.2.1 Stage-wise DSR

Stage-wise DSR 방식은 Figure 2와 같이 화자 인식을 하는 과정을 **Frontend** 단계와 **Backend** 단계로 나누어 구성한다. Frontend 단계는 어떤 발성 데이터로부터 그 화자의 특성을 나타내는 특징 벡터를 유도하는 함수라고

할 수 있고, Backend 단계는 두 개의 특징 벡터가 주어졌을 때 그 두 특징 벡터의 유사도를 측정하는 함수이다.

이런 방식으로 DSR을 나누어 구성하는 이유는 임의의 두 발성 데이터를 추출하고 그들간 유사도(거리)를 측정하여 화자 인식을 수행하기 위함이다. 어떤 발성 데이터베이스 X 에서 추출한 임의의 두 발성 데이터 x, y ($x \in X, y \in X$)가 같은 화자로부터 발생한 것인지를 판별하려면 두 발성의 유사도를 수치화해야 할 필요성이 생긴다. 만약 두 발성 데이터 간의 유사도를 “거리”를 이용하여 수치화하겠다고 한다면, 두 발성 데이터 x, y 를 일종의 n 차원 벡터로 취급할 수 있다. 그렇다면 익히 알려진 Euclidean Distance나 Cosine Distance를 이용하여 두 데이터간 거리를 측정하여 이 거리가 특정 Threshold를 만족시키는지에 따라 두 발성 데이터의 화자가 같은지를 판별할 수 있다.

그러나, 현실적으로 임의의 발성 데이터에 대하여 Euclidean Distance, Cosine Distance를 적용하여 유사도를 판별하는 것은 불가능하다. 왜냐하면 발성 데이터들이 같은 문장을 발음한 데이터라는 보장이 없고, 설령 같은 문장을 같은 사람이 발음하더라도 그 두 벡터간의 거리가 화자의 유사도를 대표해줄 수 없기 때문이다.

바로 이러한 상황 속에서, DNN을 이용하여 기존의 발성들의 화자 특징을 대표하는 특징 벡터를 추출하는 아이디어가 등장하였다. 다시 말하면, 기존의 발성 데이터들은 벡터 공간 X 에서 화자가 누군지에 관계없이 무질서하게 배열되어 있었으므로 두 데이터간 거리를 구하더라도 화자가 같은지 판별하는 기준으로 사용할 순 없었지만, 만약 발성 데이터를 거리가 유의미한 공간 E 로 사영시키는 함수 $f : X \rightarrow E$ 가 존재한다면 기존의 거리 함수를 통해 발성들의 유사도를 수치화할 수 있게 된다. 따라서 발성 데이터 x 에서 특징 벡터를 추출하는 함수 f 는 서로 같은 화자로부터 나온 발성은 거리가 가까운 특징 벡터를, 서로 다른 화자로부터 나온 발성은 거리가 먼 특징 벡터를 반환해야 한다.

바로 이러한 배경에서 Stage-wise DSR 방식의 Frontend와 Backend의 개념이 등장한다. Frontend 단계가 곧 앞에서 언급한 함수 $f : X \rightarrow E$ 에 대응되는 단계로, 임의의 발성 데이터로부터 화자 고유의 특징을 추출하여 정해진 차원의 특징 벡터를 추출하는 일을 한다. 그리고 Backend 단계는 Frontend 단계를 거친 두 특징 벡터의 유사도를 수치화하는 역할을 하므로 수치화된 유사도가 특정 기준값보다 크다면 두 발성 데이터가 같은 사람으로부터 유래했다고 판별하고 그렇지 않다면 다른 사람으로부터 유래했다고 판별할 수 있게 된다.

Frontend는 주로 DNN을 이용하여 구현하게 되고, Backend는 앞서 언급한 Euclidean Distance나 Cosine Distance 또는 PLDA를 이용하여 구현하는 경우가 많다. 이러한 Stage-wise DSR 방식을 따르는 사례는 대표적으로 **d-vector system**, **x-vector system**을 들 수 있다.

- **d-vector system**

d-vector system은 Stage-wise DSR의 Frontend를 훈련시키기 위한 방법으로, 발성으로부터 특징 벡터를 추출하기 위해 먼저 화자 식별을 위한 DNN을 먼저 훈련시키는 전략을 취한다. 이는 화자 식별을 잘 수행하는 모델이라면 가장 마지막 은닉층이 화자 고유의 정보를 담고 있을 것이라는 가정에서 출발한다. 그래서 화자 식

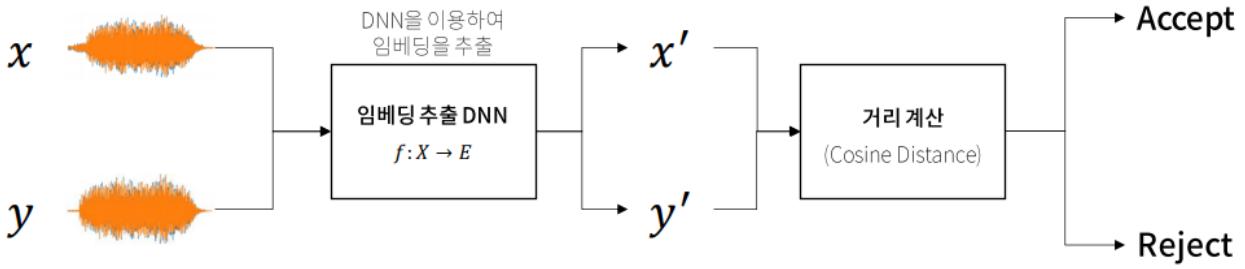


Figure 2: Stage-wise DSR 방식의 기본적인 구성. 왼편의 임베딩 추출 DNN이 Frontend를, 오른편의 거리 계산 함수가 Backend를 이룬다.

별을 위한 DNN을 먼저 훈련시킨 뒤 마지막 출력층을 제거하여 원래 가장 마지막 은닉층이었던 층의 출력을 특징 벡터로 사용한다. Figure 3가 이러한 구조를 잘 보여준다. [7]

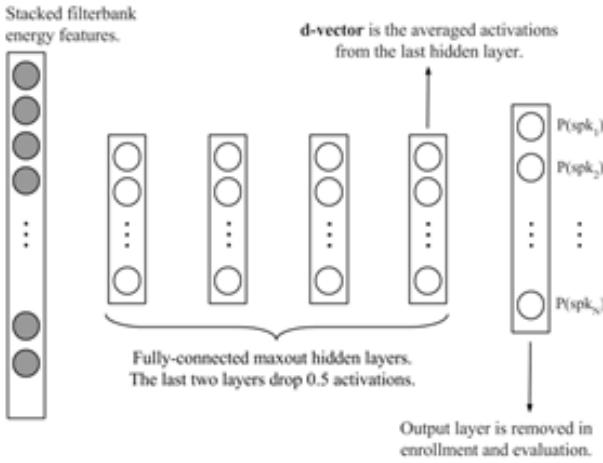


Figure 3: d-vector system에서 사용하는 DNN.

이러한 d-vector system은 논문이 발표되었을 당시 state-of-the-art였던 i-vector system에 준하는 성능을 보여주었고, 특히 Noise에 robust한 특성, False rejection이 낮을 때 false acceptance가 낮은 특성 등을 보이면서 i-vector system보다 큰 잠재성을 보여주었다. DNN을 화자 인식에 사용한다는 점이 연구의 동기가 되어 이후 x-vector system과 같은 더 향상된 모델이 등장하기도 하였다.

• x-vector system

x-vector system 역시 모델을 화자 식별을 위해 먼저 훈련시킨 뒤 은닉층의 출력을 특징 벡터로 이용한다는 점에서 d-vector와 유사하고, d-vector의 모델을 개선한 것에 가깝다. Figure 4가 x-vector의 모델 구조를 보여주는데, 모델은 24차원의 filterbank 데이터를 입력으로 받고, 5개의 time-delay layer가 frame-level의 정보를 추출한다. 이렇게 추출된 정보는 statistical pooling layer의 입력이 되고, 그 이후부터는 segment-level로 화자 식별을 수행한다. 화자 식별을 위한 훈련이 충분히 진행되고 나면 출력층 이전의 두개의 은닉층에서 특징벡터를 추출하여 Backend 단계로 넘긴다. [5]

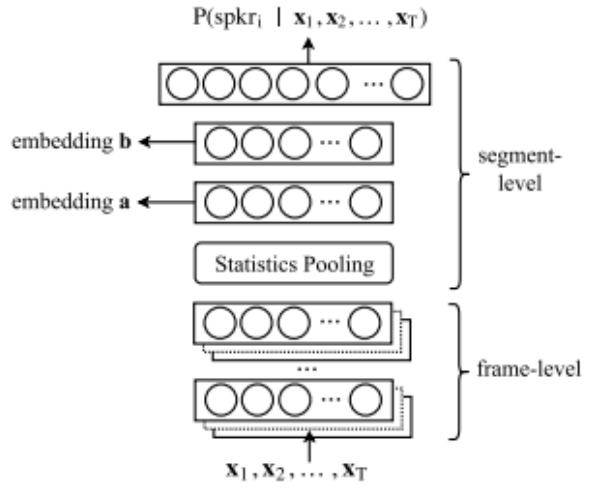


Figure 4: x-vector system에서 사용하는 DNN.

Data augmentation 기법과 함께 훈련시켰을 때 x-vector system은 i-vector system은 물론 d-vector system 보다도 낮은 EER을 보여주었다.

- Task Mismatch Problem: d-vector와 x-vector의 약점

d-vector system과 x-vector system이 i-vector system을 준하거나 더 높은 성능을 얻었음에도 불구하고, 시스템의 구조적 특성에 따른 문제점이 존재하였는데, 이는 학습 단계에서의 Accuracy와 평가 단계에서의 Accuracy에 Performance Gap이 발생한다는 것이었다. 이는 학습과 평가 과정에서 Task Mismatch Problem이 발생하였기 때문인데, 특징 벡터를 추출하는 DNN을 학습시키기 위해 화자 식별을 학습시킨다는 점에서 학습시킨 것과 실제로 활용하는 것이 맞지 않는다는 것이다. 이러한 문제는 네트워크가 훈련된 이후에 화자 식별에 등록되지 않은 화자의 발성을 입력하였을 때 화자 인증의 성능이 하락하는 문제를 야기하였다. 화자 식별에 등록되지 않은 화자를 unseen speaker라고 말하기도 하는데, 이러한 unseen speaker에 대한 특징 벡터 추출 성능을 높이기 위해 deep metric learning이나 end-to-end DSR이 연구되고 있다.

1.2.2 End-to-end DSR

End-to-end DSR 방식은 Stage-wise 방식처럼 화자 인식의 과정을 분할하는 것이 아니라 입력 데이터로부터 곧바로 화자 특징 벡터를 추출한다. 기존의 Stage-wise 방식은 Frontend와 Backend를 별도로 설계하고 역전파하여야 했지만, End-to-end 방식은 그 과정이 하나로 연결된 단일 Loss function을 사용하므로, 전체 구조가 동시에 훈련된다. 전체 구조를 한번에 훈련시켜야 하므로 훈련 데이터가 상당히 많이 요구되고, loss function을 설계자가 직접 고안해야 한다는 한계점이 존재한다. [1]

그럼에도 불구하고, Deep metric learning과 meta learning을 통해 높은 성능을 얻은 사례가 존재하고, RawNet과 같이 테스크에 맞게 시스템을 효율적으로 설계하여 높은 성능을 보인 사례가 있다. 또한 오늘날 모델 훈련에 사용할 수 있는 데이터의 양과 질이 상승하고 있으므로 데이터 부족으로 나타나는 문제점 역시 해소되고 있다.

- Triplet Network

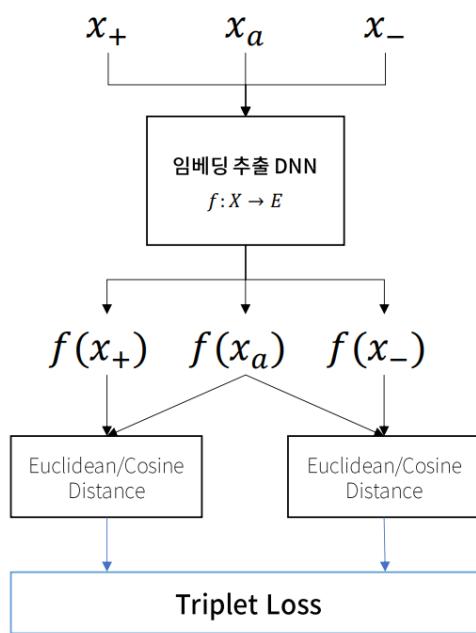


Figure 5: Triplet network에서 사용하는 DNN의 구조. 끝 단에서 계산된 loss가 함수 f 까지 역전파되어 전체 구조가 한번에 훈련된다.

Triplet Network는 기존의 d-vector system처럼 임베딩 추출 DNN을 간접적인 방법으로 학습시키는 것이 아니라 같은 화자의 데이터를 클러스터링 시키도록 모델을 직접 학습시키는 방법이다. 기존의 d-vector나 x-vector 기법은 화자 식별을 먼저 훈련시킨 뒤 특징 벡터를 그 모델의 은닉층에서 간접적으로 추출하는 방식이었지만 triplet network는 최종 출력층에서 나오는 벡터가 곧 특징 벡터가 된다. 이러한 triplet network의 궁극적인 목표는 같은 종류의 데이터들에서 유래한 특징 벡터들이 가깝게 배치되도록 클러스터링 하는 것이다. [9]

훈련시에는 데이터베이스로부터 3가지 데이터(Triplet)를 추출하는데, 임의의 발성 x_a , x_a 와 화자가 같은 발성

x_+ , x_a 와 화자가 다른 발성 x_- 이 그에 해당된다. 이렇게 triplet을 추출하였다면 아래와 같은 방식으로 loss를 계산한다.

$$\Delta = d(f(x_a), f(x_+)) - d(f(x_a), f(x_-))$$

$$\mathcal{L} = \sum \max(0, \Delta + \alpha)$$

- \mathcal{L} : triplet loss 값
- x_a : 임의의 데이터
- x_+ : x_a 와 화자가 같은 데이터
- x_- : x_a 와 화자가 다른 데이터
- f : 특징 벡터를 추출하는 함수
- d : 두 특징 벡터간 거리를 측정하는 함수
- Δ : 거리의 차
- α : Margin 값 (hyperparameter)

여기에서 \mathcal{L} 는 모델이 학습한 정도를 알 수 있는 척도가 되며, \mathcal{L} 값이 작아지도록 역전파를 수행한다. \mathcal{L} 가 작아지게 하는 것은 $d(f(x_a), f(x_+))$ 는 크게 하되 $d(f(x_a), f(x_-))$ 는 크게 하는 것과 같으므로 결과적으로 화자가 같은 발성 데이터의 특징 벡터들은 서로의 거리가 가깝게 배치되고, 화자가 다른 발성 데이터의 특징 벡터들은 거리가 멀게 배치된다. [9]

이러한 Triplet Network(Triplet Loss)를 통해 이전의 d-vector가 보였던 Task Mismatch Problem을 해결할 수 있었지만 Triplet Loss 또한 손실함수의 설계 자체에 결함이 존재함이 밝혀졌다. Triplet Loss는 직관적으로 보기에는 같은 화자가 발음한 발성 데이터는 클러스터링 될 것으로 보이지만 실제로는 각 특징 벡터들을 하나의 클러스터로 묶을 것이라는 보장이 되지 않는다. Figure 6이 그 점을 아주 잘 보여주는데, 같은 종류의 클러스터(보라색) 사이에 다른 종류의 특징 벡터(노란색)가 위치하면, 손실 함수를 계산한 뒤 역전파를 수행해도 노란색 특징 벡터와 보라색 특징 벡터의 거리가 멀어지도록 모델의 파라미터를 수정하게 되므로 결과적으로 보라색 특징 벡터는 모이지 못한다. [6]

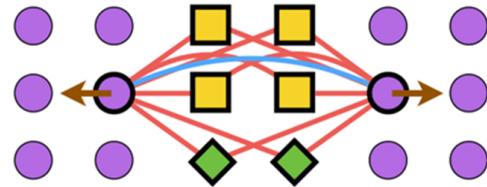


Figure 6: Triplet loss로 같은 종류의 특징 벡터가 클러스터링 되지 못한 상황을 묘사한 그림

또한 컴퓨팅 파워 측면에도 문제점이 존재하는데, Triplet Loss를 계산하기 위해서는 이론상 모든 Triplet을 확인해야 하므로, 상당히 큰 컴퓨팅 파워가 요구되게 된다.

- Prototypical Network

Prototypical Network 역시 Triplet Network와 같이 특징 벡터 추출 DNN을 간접적인 방법으로 학습시키는 것이 아니라 같은 화자의 데이터를 클러스터링 시키도록 모델을 직접 학습시키는 방법이다. 그런데 두 기법에는 큰 차이점이 있는데, 과거의 triplet network에서는 triplet loss를 계산하기 위해 모든 triplet을 확인해야 했다면 prototypical network loss는 한 화자의 평균 벡터(Centroid 또는 Prototype)과만 비교한다. [8]

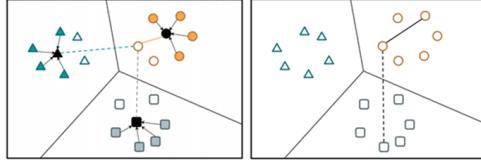


Figure 7: Prototypical network(좌측)와 triplet network(우측)의 loss 계산 방법을 비교한 그림. Triplet network는 모든 triplet에 대하여 loss를 계산하지만, prototypical network는 각 클러스터들의 평균 벡터와만 손실을 계산한다.

또한 prototypical network는 **meta-learning** 기법을 이용하여 훈련되는데, meta-learning은 훈련 데이터에서 찾을 수 없는 unseen class에 대해서도 높은 성능을 보장하기 위해서 고안되었다. 이러한 meta-learning은 훈련 데이터가 적더라도 충분히 학습할 수 있는 잠재성을 가지고 있기 때문에 오늘날 현실의 대부분의 문제 상황에 적합하다. meta-learning 두 가지 데이터 셋, support set $(x, y) \in \mathcal{S}$ 과 query set $(x, y) \in \mathcal{Q}$ 으로 훈련 및 시험된다. Support set에 포함된 class의 수가 C 개이고, 각 class마다 K 개의 샘플 데이터가 있을 때, 이러한 상황에서의 meta-learning을 특별히 K -shot C -class 태스크라고 한다.

이제 meta-learning의 관점에서 Prototypical network를 분석하면, prototypical network loss를 논의하는 것이 가능하다. 우선 k 번째 화자에 대한 support set을 $S_k = \{(x_i, y_i)\}_{i=1}^N$ 라고 하였을 때 각 화자의 centroid(또는 prototype)은 아래 식으로 계산된다.

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f(x_i)$$

- S_k : k 번째 화자에 대한 support set
- N : S_k 에 포함된 원소의 수
- (x_i, y_i) : support set의 i 번째 입력 데이터와 레이블
- c_k : k 번째 화자에 대한 centroid
- f : 특징 벡터를 추출하는 함수

그리고 prototypical network의 손실함수인 prototypical network loss는 이러한 centroid가 서로 멀리 떨어지도록 학습시키는데, 미지의 데이터가 어느 분류에 속하는지 결정할 때는 그 데이터의 특징 벡터가 어느 centroid에 가장 가까운지를 판별하게 된다.

$$p(y = y_i | x_i) = \frac{\exp(-d(f(x_i), c_y))}{\sum_j \exp(-d(f(x_i), c_j))}$$

$$\mathcal{L} = \sum_{(x_i, y_i) \in Q} -\log p(y = y_i | x_i)$$

- Q : Query set
- c_k : k 번째 화자에 대한 centroid
- (x_i, y_i) : Query set의 i 번째 입력 데이터와 레이블
- f : 특징 벡터를 추출하는 함수
- d : 거리 함수

prototypical network는 triplet network에 비해 여러 장점을 가지고 있는데, loss를 계산하기 위해 오직 centroid만 확인해도 충분하므로 triplet network보다 요구되는 컴퓨팅 파워가 적고, triplet network는 sampling strategy나 margin 등 정해야 하는 hyperparameter가 많은 편이지만 prototypical network는 정해야 하는 hyperparameter가 없다는 장점이 있다.

1.3 화자 인식을 위한 데이터 증강

데이터 증강(Data augmentation)은 오늘날 효율적인 네트워크를 학습시키는데 필수적인 과정이다. 데이터 증강은 기존에 사용할 수 있었던 데이터 셋을 이용하여 원본 데이터 셋에 없는 데이터를 생성하여 학습에 사용함으로써 네트워크의 일반화 성능을 끌어올리는 기법이다.

Computer vision 영역에서의 State-of-the-art 객체 식별 네트워크들은 이미지를 회전하거나 확대/축소하는 방법으로 데이터를 증강하여 학습에 사용하는 모델이 많고, 소리 데이터를 사용하는 화자 인식이나 Acoustic Scene Classification 등의 영역에서도 태스크에 알맞은 데이터 증강에는 어떤 기법이 있는지 활발히 연구되고 있다.[4]

1.3.1 Naïve Methods

소리 데이터에 대하여 가장 쉽게 떠올릴 수 있는 데이터 증강은 Pitch, Time, Loudness를 조절하거나, Dropout이나 Gaussian noise를 적용하는 것이 존재할 것이다. Jan Schlüter et al.은 Deep neural network를 이용하여 Singing voice detection을 수행하는 태스크에서 이런 단순 데이터 증강이 어떤 효과를 보이는지 조사하였다.[4]

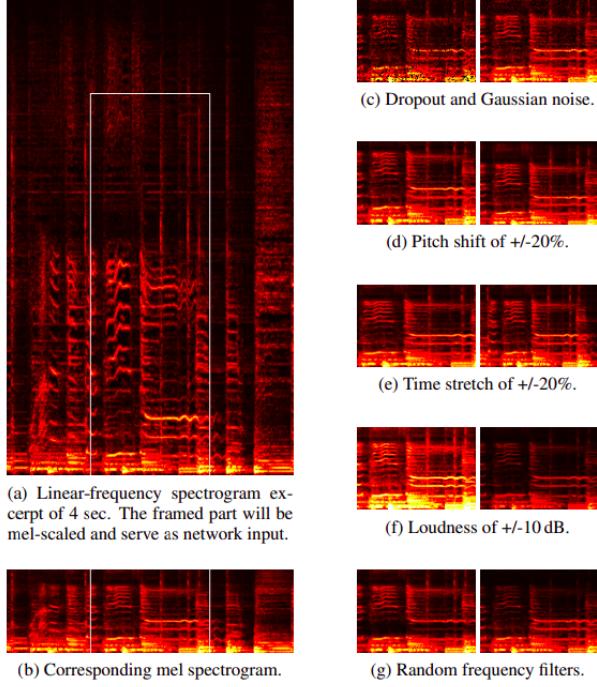


Figure 8: Spectrogram을 통한 데이터 증강 기법들의 시각화

In-House A 데이터 셋과 In-House B 데이터 셋으로 평가한 결과가 Figure 9에 정리되어 있다. 결과를 요약하면, **Pitch shifting**과 **Random frequency filter**가 다른 증강 기법에 비해 가장 큰 성능 향상을 가져왔고, **Time stretching**은 성능을 하락시키진 않으나 크게 향상시키지도 않았으며, 나머지 기법들은 비효율적이거나 오히려 데이터를 손상시킴이 드러났다.

Pitch shifting과 **Random frequency filter**가 training과 test-time 전반에 큰 영향력을 보여주었는데, 이를 이용하면 pitch 변화에 robust한 classifier를 디자인할 수 있을 것으로 보인다.

1.3.2 SpecAugment

SpecAugment는 Speech recognition task에 적용할 수 있는 간단한 데이터 증강 기법으로, 해당 논문이 발표된 2019년 당시 Speech recognition을 위한 데이터 증강 기법들 중 State-of-the-art를 달성한 기법이다. SpecAugment는 Spectrogram 그 자체에 바로 적용되며 Time warping, Frequency masking, Time masking 세 가지 방법을 제안한다.[3]

- **Time warping**

Time warping은 입력으로 주어진 Spectrogram의 중앙을 가로로 지나는 축에서 임의의 점을 선택한 뒤 해당 선을 따라 왼쪽 또는 오른쪽으로 w 만큼 웨곡시키는 방법이다. 여기에서 w 는 0부터 time warp parameter W 사이에서 균일 확률 분포에 따라 선택된다. 실제 구현할 때는 Tensorflow의 sparse_image_warp 함수를 이용하여 구현한다.

- **Frequency masking**

Frequency masking은 Spectrogram의 frequency 대역 중 일부 연속된 구간을 선택하여 해당 구간을 0 또는 평균으로 채우는 방법이다. 이는 v 를 주파수 채널의 수, F 를 Frequency masking parameter, f 를 $(0, F)$ 구간에서 균일한 확률로 선택한 값, f_0 를 $[0, v - f]$ 구간에서 균일한 확률로 선택한 값이라고 했을 때 Frequency channel $[f_0, f_0 + f)$ 를 masking 하는 것과 같다.

- **Time masking**

Time masking은 Spectrogram의 Time 대역 중 일부 연속된 구간을 선택하여 해당 구간을 0 또는 평균으로 채우는 방법이다. 이는 τ 를 시간 채널의 수, T 를 Time masking parameter, t 를 $(0, T)$ 구간에서 균일한 확률로 선택한 값, t_0 를 $[0, \tau - t]$ 구간에서 균일한 확률로 선택한 값이라고 했을 때 Time channel $[t_0, t_0 + t)$ 를 masking 하는 것과 같다.

그리고 위에서 제안한 방법들을 유한번 사용하여 구성한 Augmentation policy는 아래 Table과 같다. 아래 표에서 m_F 는 Frequency masking을 적용하는 횟수, m_T 는 Time masking을 적용하는 횟수이다.

Policy	W	F	m_F	T	p	m_T
None	0	0	-	0	-	-
LB	80	27	1	100	1.0	1
LD	80	27	2	100	1.0	2
SM	40	15	2	70	0.2	2
SS	40	27	2	70	0.2	2

Figure 10: SpecAugment의 Augmentation policy

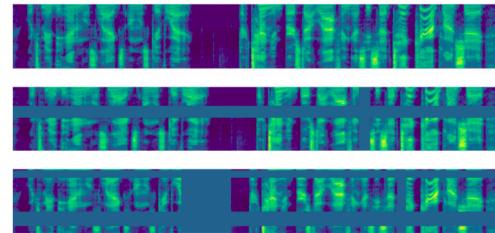


Figure 11: SpecAugment의 Augmentation policy를 적용한 모습. 가장 위 부터 None, LB, LD policy를 적용한 것임.

이러한 SpecAugment는 Overfitting 문제를 Underfitting 문제로 변환한다. 즉, 모델이 데이터 셋에 과적합되는 것을 방지하고, unseen data에 대한 일반화 성능을 끌어올린다. 제안된 3가지 기법들 각각을 살펴보면 Masking 기법은 위에서 설명한 것과 같이 일반화 성능을 크게 끌어올릴 뿐만 아니라 Spectrogram의 일부를 특정 값으로 채워버리는 것에 불과하기 때문에 요구하는 Computing power가 상당히 적다는 큰 장점이 있다. 그런데 Time warping의 효과는 앞에서 다룬 두 방법보다는 작은 것으로 평가되는데, 일반화 성능에 기여함에도 불구하고 세 가지 기법들 중 가장 많은 Computing power를 요구하기 때문이다.

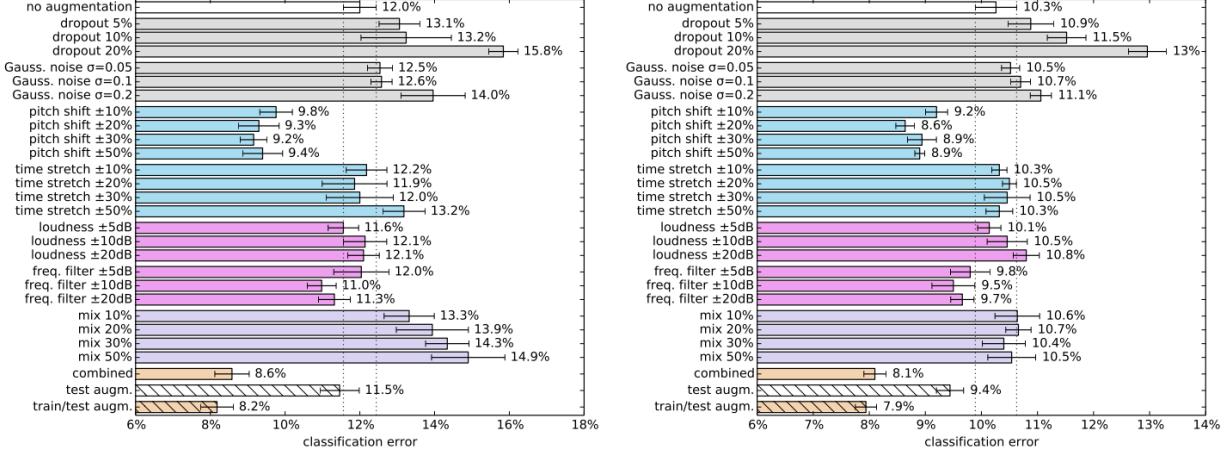


Figure 9: 서로 다른 데이터 셋으로 평가한 각 데이터 증강의 성능. (왼쪽: In-House A, 오른쪽: In-House B) 5번 반복 실험하여 평균과 95% 신뢰구간을 측정하였다.

2 진행한 실험

이 섹션에서는 그동안 연구실에서 화자 인식 모델을 설계 및 실험한 과정/결과를 간단하게 서술한다.

2.1 진행한 실험의 개요

연구실에 참여하기 시작한 2020년 12월, 5층 fully-connected 네트워크를 사용한 d-vector 논문을 읽고 이를 PyTorch로 구현하는 것으로 첫 실험을 시작하였고, 첫번째 실험을 마친 뒤에는 은닉층에서 화자의 특징 벡터를 추출한다는 d-vector의 기본 Concept는 유지한채로 Convolutional layer, Max/Average Pooling, Res-Block 등을 차용하여 성능을 EER 6.5%까지 끌어올리는 것을 목표로 하였다. Dataset은 초기에는 RSR 2015를 사용하였으나 ResBlock을 이용한 이후에는 VoxCeleb 1을 이용하였다.

은닉층(4층)의 출력 벡터를 임베딩으로 하였다.

2.1.1 Fully Connected Model

실험의 첫 걸음마로 작성한 모델은 5층 Fully Connected Model이다. 당시에는 인공지능 모델 뿐만 아니라 화자 인식 학습을 위한 개발 환경 조성 부터 기본적인 데이터 가공 방법까지 부족한 부분이 많았으므로, 실험하는 네트워크 모델은 최대한 간단한 모델이어야 했다.

1. Dataset: RSR2015

전체 Data 중 모든 데이터를 사용하지 않고 그 중 일부를 무작위 추출하여 196명의 화자, Train data(약 10,000 쌍), Validation data(약 200,000 쌍), Test data(약 300,000 쌍) 목록을 생성하였다.

2. 음성 특징 추출 방법

13차원의 MFCCs를 추출하였고, 각 음성을 200 time-step의 window 들로 분할하여 사용하였다. 즉, 음성의 길이가 서로 다르더라도 window 단위로 학습을 진행하기 때문에 입력차원수가 고정적 인 모델에서도 사용할 수 있었다.

3. 화자 임베딩 추출 방법

5층 Fully Connected 네트워크에서 가장 마지막

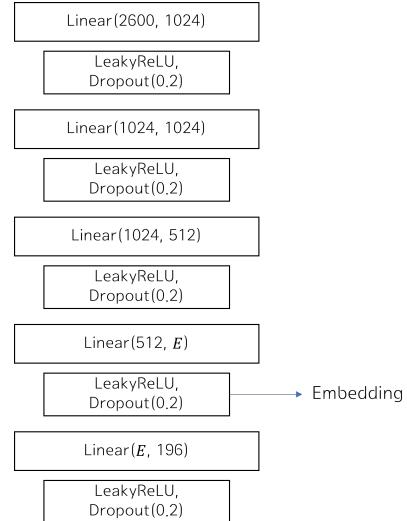


Figure 12: Fully Connected Model의 개형.

위와 같이 실험 환경을 준비하고, 가장 마지막 은닉층의 차원 수(E)를 조정해보면서 성능 변화를 알아보았다.

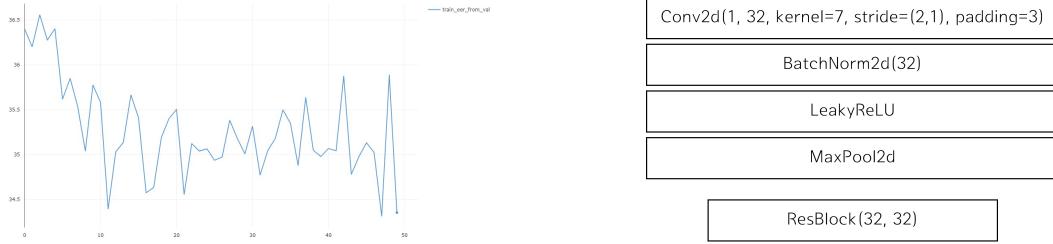


Figure 13: Fully Connected Model에서 임베딩의 차원수를 128로 설정했을 때의 EER Graph. 최종 EER은 34.35.

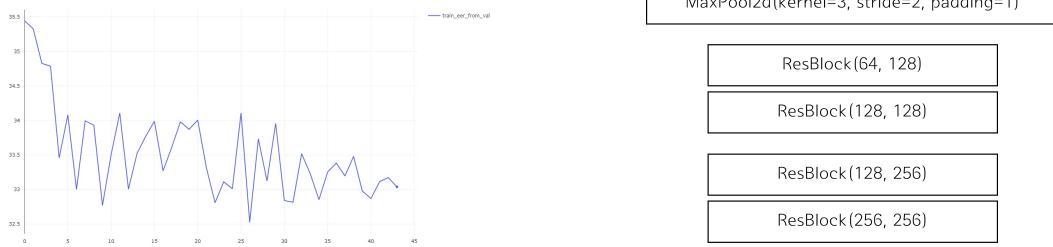


Figure 14: Fully Connected Model에서 임베딩의 차원수를 64로 설정했을 때의 EER Graph. 최종 EER은 33.04.

결과적으로 임베딩의 차원수를 64로 설정한 것이 128로 설정한 것보다 더 적은 EER을 보였으므로 임베딩의 크기를 줄이는 것이 화자의 특징을 더 잘 요약한다고도 말할 수 있겠으나, 기본적으로 30이상의 EER은 화자 분별을 거의 못하고 있다고 말해도 과하지 않은 EER이므로, 임베딩의 크기가 화자 특징에 어떤 영향을 주는지를 논하기 어려운 면도 있었다.

2.1.2 ResNet과 Adaptive Average/Max Pooling

Fully Connected Model에 대한 성능 튜닝을 마무리하고 Residual Block에 대한 공부를 한 뒤 이를 응용하여 화자 인식 모델을 설계하였다. 기존의 d-vector가 제안한 것 처럼 모델의 마지막 은닉층에서 임베딩을 추출하여 Identification/Verification에 사용한다는 개념은 동일하나, Residual Block과 Pooling을 포함한다는 점에서 차이점이 있다.

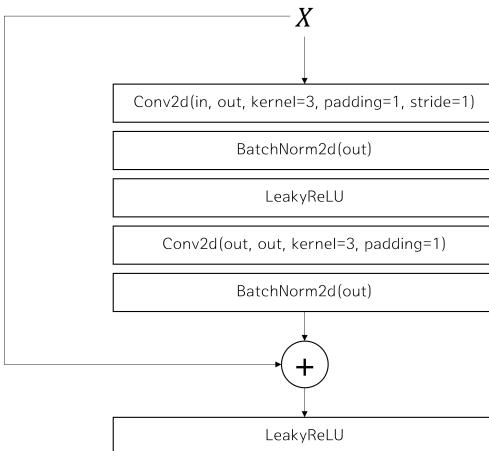


Figure 15: 실험에 사용한 ResBlock의 개형.

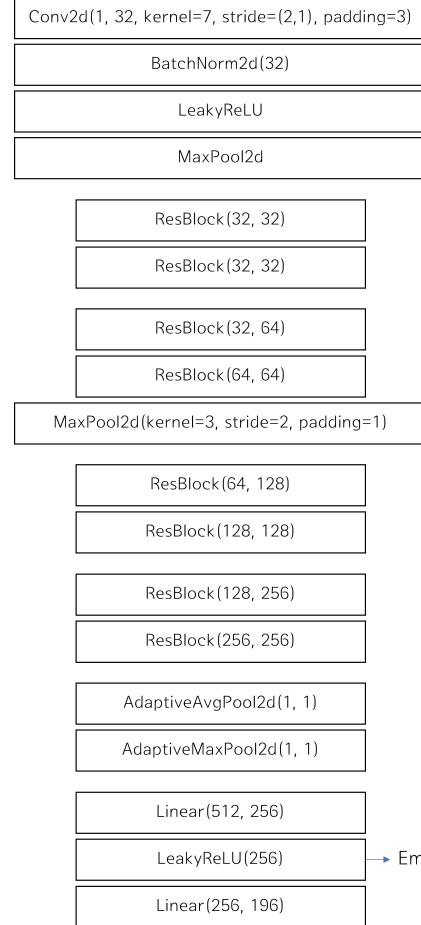


Figure 16: 설계한 ResNet의 개형. 최종 EER은 10.42.

위와 같은 모델로 실험했을 때의 EER은 10.42로, 실험한 다른 모델들 중 가장 좋은 성능을 보였다. Residual Block으로 인해 학습 속도가 개선되었고, Batch Normalization이 Overfitting을 방지하는 효과를 가져온 것으로 보인다.

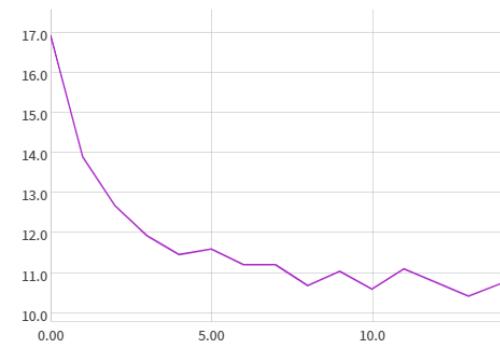


Figure 17: 설계한 ResNet의 EER Graph. 최종 EER은 10.42.

References

- [1] Tobias Glasmachers. *Limits of End-to-End Learning*. 2017. arXiv: [1704.08305 \[cs.LG\]](https://arxiv.org/abs/1704.08305).
- [2] Abu Quwsar Ohi et al. “Deep Speaker Recognition: Process, Progress, and Challenges”. In: *IEEE Access* 9 (2021), pp. 89619–89643. DOI: [10.1109/ACCESS.2021.3090109](https://doi.org/10.1109/ACCESS.2021.3090109).
- [3] Daniel S. Park et al. “SpecAugment: A Simple Augmentation Method for Automatic Speech Recognition”. In: *INTERSPEECH*. 2019.
- [4] Jan Schlüter and Thomas Grill. “Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks”. In: *ISMIR*. 2015.
- [5] David Snyder et al. “Deep Neural Network Embeddings for Text-Independent Speaker Verification”. In: *Proc. Interspeech 2017*. 2017, pp. 999–1003. DOI: [10.21437/Interspeech.2017-620](https://doi.org/10.21437/Interspeech.2017-620). URL: <http://dx.doi.org/10.21437/Interspeech.2017-620>.
- [6] Hyun Oh Song et al. *Deep Metric Learning via Facility Location*. 2017. arXiv: [1612.01213 \[cs.CV\]](https://arxiv.org/abs/1612.01213).
- [7] Ehsan Variani et al. “Deep neural networks for small footprint text-dependent speaker verification”. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 4052–4056. DOI: [10.1109/ICASSP.2014.6854363](https://doi.org/10.1109/ICASSP.2014.6854363).
- [8] Jixuan Wang et al. *Centroid-based deep metric learning for speaker recognition*. 2019. arXiv: [1902.02375 \[cs.LG\]](https://arxiv.org/abs/1902.02375).
- [9] Chunlei Zhang and Kazuhito Koishida. “End-to-End Text-Independent Speaker Verification with Triplet Loss on Short Utterances”. In: *Proc. Interspeech 2017*. 2017, pp. 1487–1491. DOI: [10.21437/Interspeech.2017-1608](https://doi.org/10.21437/Interspeech.2017-1608). URL: <http://dx.doi.org/10.21437/Interspeech.2017-1608>.
- [10] 서영주 김회린. “최근 화자인식 기술 동향”. In: *전자공학회지* 41.3 (2021), pp. 40–49. ISSN: 1016-9288.