LG 전자

음성지능 - 화자인식 및 음성합성

[화자인식 실습]

TA : 정영문
Advisor : 김회린 교수님
KAIST SSSC Lab

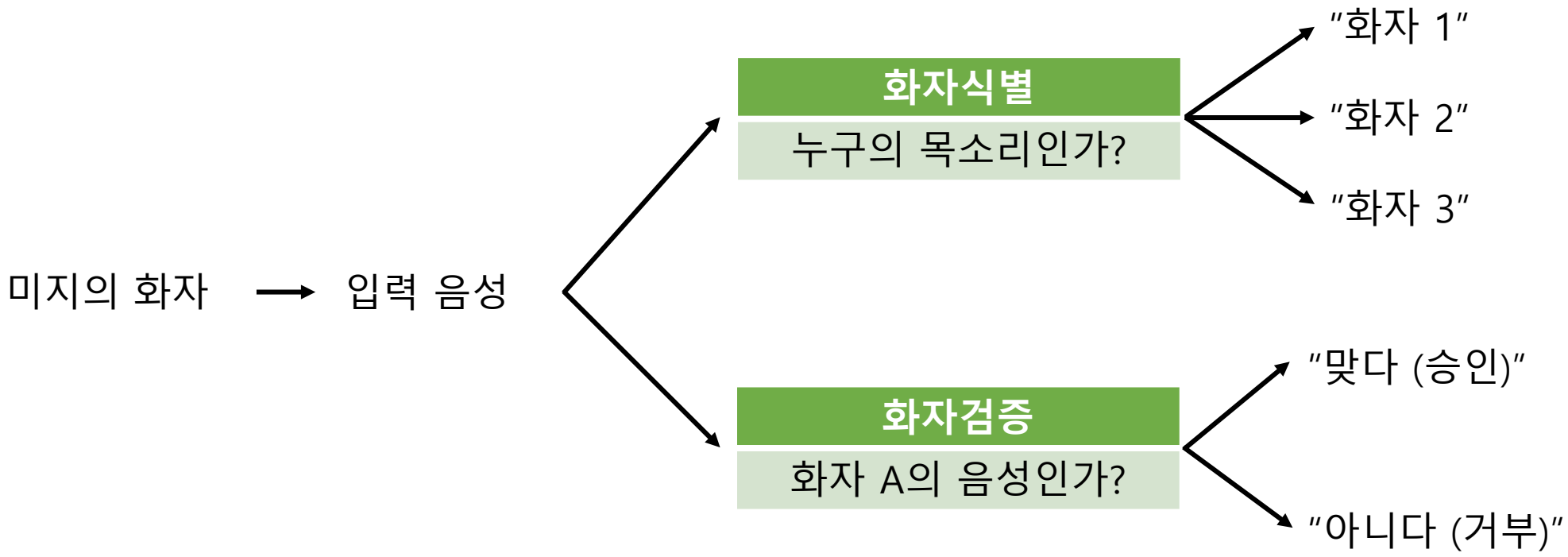
[Contact Info.]
정영문 : dudans@kaist.ac.kr

실습 순서

- 개념 설명
  - 화자인식
  - d-vector 방식
- 실습 설명
  - 훈련
  - 등록
  - 테스트
    - 화자식별
    - 화자검증
- 개별 실습

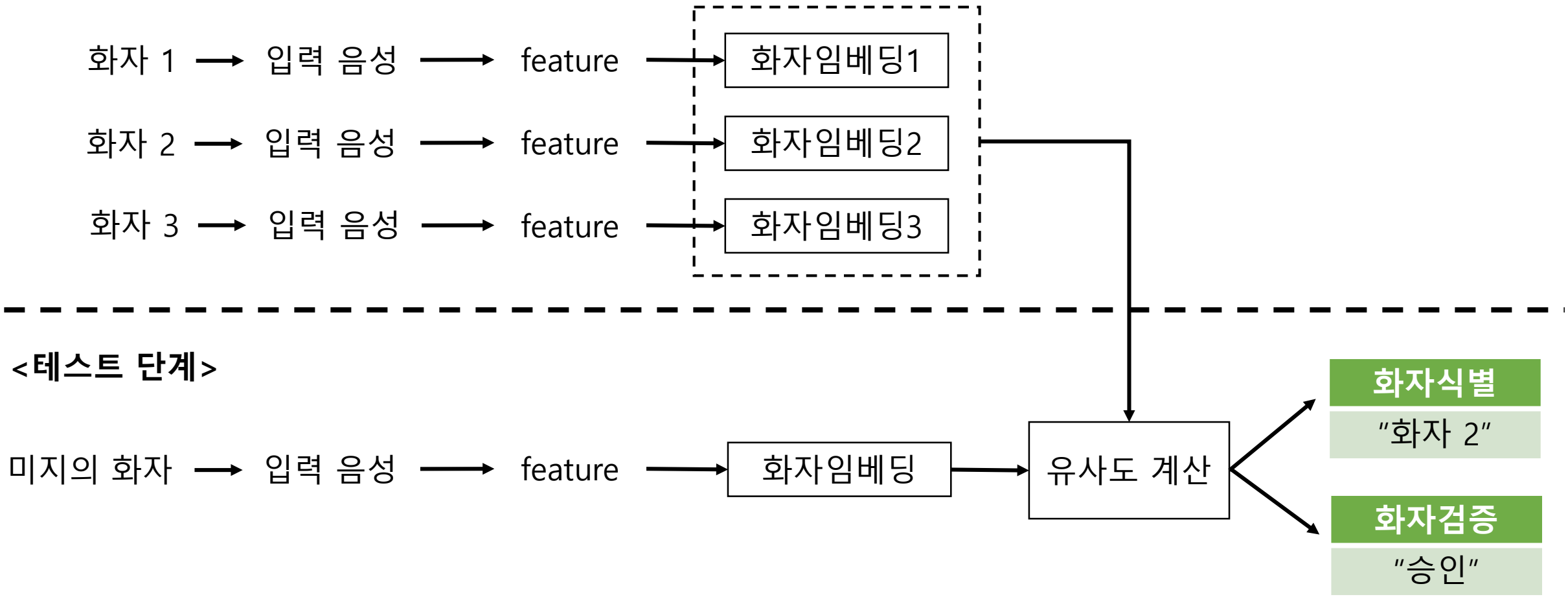
화자인식 (speaker recognition)

- 화자식별 (speaker identification) vs 화자검증 (speaker verification)



화자인식 과정

<등록 단계>

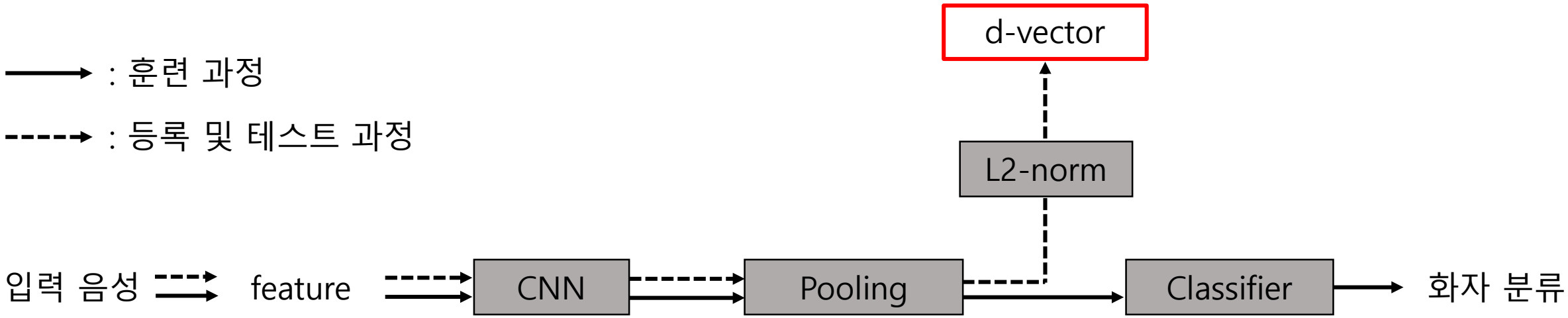

화자 1
입력 음성
feature
화자 2
입력 음성
feature
화자 3
입력 음성
feature
화자임베딩1
화자임베딩2
화자임베딩3
<테스트 단계>
미지의 화자
입력 음성
feature
화자임베딩
유사도 계산
화자식별
"화자 2"
화자검증
"승인"

d-vector 방식

- **특징**
  - 딥러닝 모델 (DNN, CNN, LSTM)을 이용
  - 화자 정보를 나타내는 **화자임베딩 (speaker embedding)**을 추출

- **추출 방법**
  - 화자분류기 (speaker classifier) 훈련
  - 훈련이 끝난 화자분류기의 마지막 hidden layer activation 값을 이용하여 d-vector 추출

CNN 기반의 d-vector 방식



- MFCC
- **Fbank**
- Spectrogram

- VGGNet
- **ResNet**

- **Average pooling**
- Spatial pyramid pooling
- Learnable dictionary encoding
- LSTM

Fully-connected layer with
- **Cross entropy loss**
- Center loss
- Angular softmax loss

화자인식 실습

- 코드 및 DB

git clone https://github.com/jymsuper/SpeakerRecognition_tutorial

  - 코드
    - Pytorch 기반 (v1.0.0, python 3)
    - pandas 라이브러리의 dataframe 자료구조를 이용하여 데이터 로드
  - 필요한 라이브러리
    - python 3.5+, pytorch, pandas, numpy, pickle, matplotlib
    - pip 혹은 가상환경 (anaconda,...)을 이용하여 설치

화자인식 실습

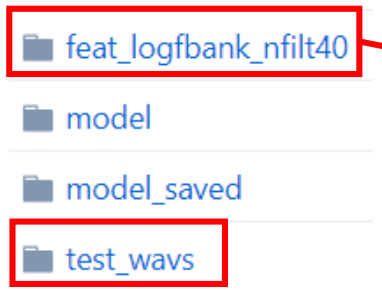
- 코드 및 DB

git clone https://github.com/jymsuper/SpeakerRecognition_tutorial

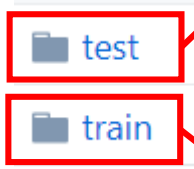
  - DB
    - 과제를 통해 수집된 DB를 이용 (clean 환경)
      - SNS단문 낭독 음성 DB (ETRI 낭독체)
      - 1m 거리의 무잡음 음성, 0도 방향, 16kHz, 16bits
    - 훈련 : 240명 화자, 각 화자 당 100개의 파일
      - Feature (log mel filterbank energy feature)만 업로드
    - 테스트 : 10명의 화자, 각 화자 당 2개의 파일
      - 각각 등록 및 테스트 용, 15초 분량
      - wav 파일 및 feature 모두 업로드
    - python_speech_features 라이브러리를 이용하여 feature 추출

폴더 설명

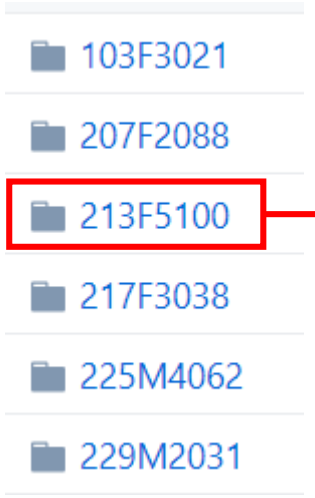
훈련 및 테스트 feature



테스트 wav 파일

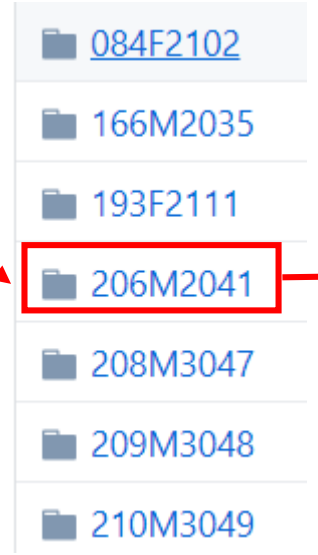


10개의 폴더
(10명의 화자)



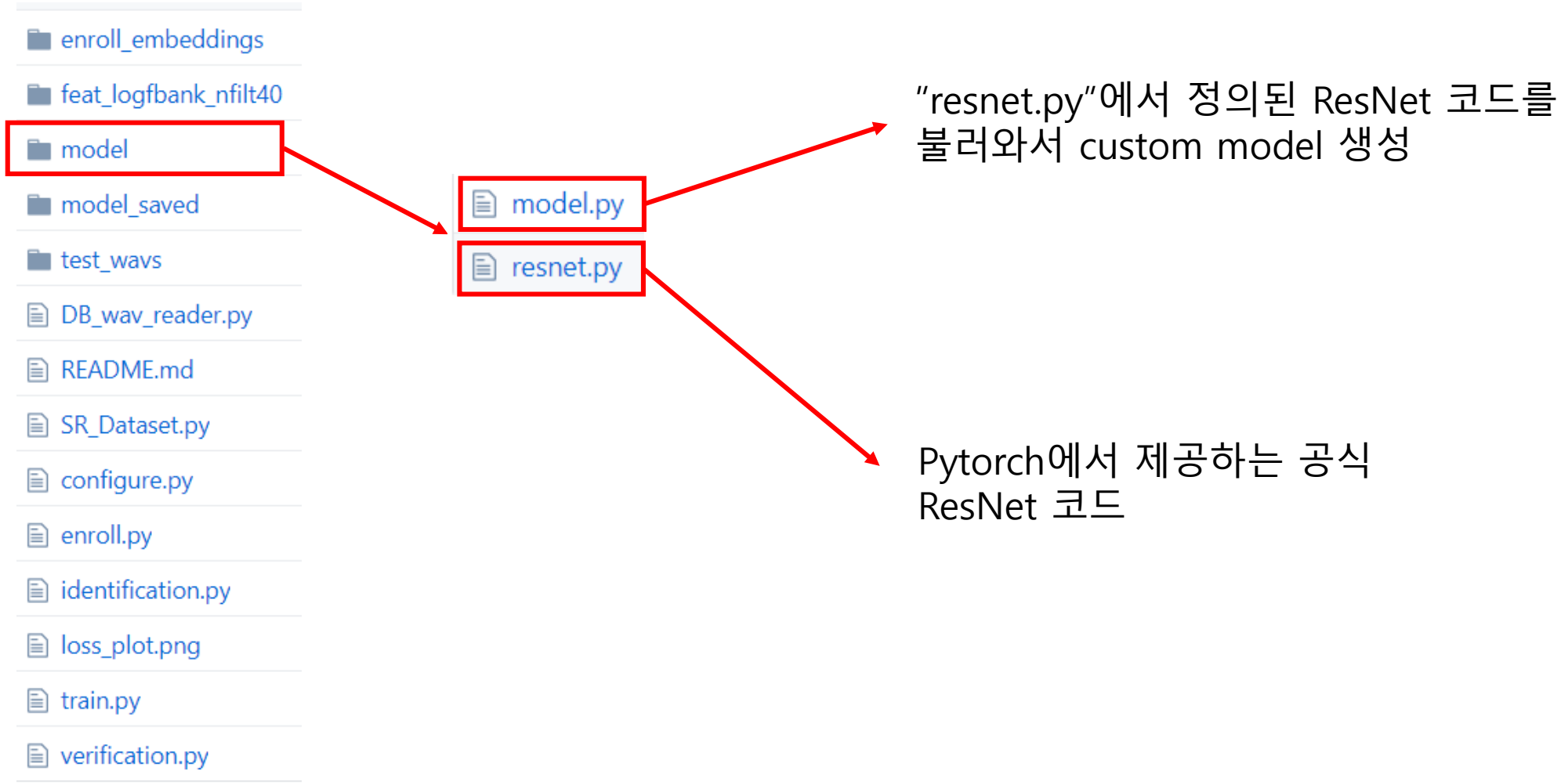
각 폴더 당 2개의 파일
(각각 등록 및 테스트 용)

240개의 폴더
(240명의 화자)



각 폴더 당 100개의 파일

폴더 설명


enroll_embeddings
feat_logfbank_nfilt40
model
model_saved
test_wavs
DB_wav_reader.py
README.md
SR_Dataset.py
configure.py
enroll.py
identification.py
loss_plot.png
train.py
verification.py
model.py
resnet.py
"resnet.py"에서 정의된 ResNet 코드를 불러와서 custom model 생성
Pytorch에서 제공하는 공식 ResNet 코드

“resnet.py”

```python
class ResNet(nn.Module):

    def __init__(self, block, layers, num_classes=1000, in_channels=1):
        self.inplanes = 16
        super(ResNet, self).__init__()
        self.conv1 = nn.Conv2d(in_channels, 16, kernel_size=7, stride=1, padding=3,
                               bias=False) # ori : stride = 2
        self.bn1 = nn.BatchNorm2d(16)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)
        self.layer1 = self._make_layer(block, 16, layers[0])
        self.layer2 = self._make_layer(block, 32, layers[1], stride=2)
        self.layer3 = self._make_layer(block, 64, layers[2], stride=2)
        self.layer4 = self._make_layer(block, 128, layers[3], stride=2)
        self.avgpool = nn.AvgPool2d(1, stride=1)
        self.fc = nn.Linear(128 * block.expansion, num_classes)
```

conv layer의 channel 개수

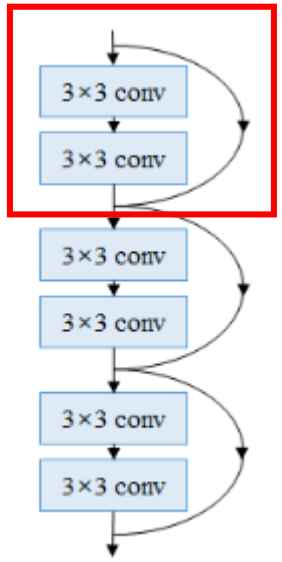
resnet-18, 34, 50, 101, 152

```python
def resnet18(pretrained=False, **kwargs):
    """Constructs a ResNet-18 model.
    Args:
        pretrained (bool): If True, returns a model pre-trained on ImageNet
    """
    model = ResNet(BasicBlock, [2, 2, 2, 2], **kwargs)
    if pretrained:
        model.load_state_dict(model_zoo.load_url(model_urls['resnet18']))
    return model
```

- 각각 layer1, 2, 3, 4에 대응됨
- Residual block의 개수
  layer1에서 block 2개, layer2에서 block 2개,
  layer3에서 block 2개, layer4에서 block 2개
- 각 block은 2개의 conv layer로 구성됨
- 총 layer 개수
  conv1 1개 + layer1 4개 + layer2 4개 +
  layer3 4개 + layer4 4개 + FC layer 1개 = 18개

“resnet.py”

“BasicBlock”



2 conv layers
+
Residual connection

```python
class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, inplanes, planes, stride=1, downsample=None):
        super(BasicBlock, self).__init__()
        self.conv1 = conv3x3(inplanes, planes, stride)
        self.bn1 = nn.BatchNorm2d(planes)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = conv3x3(planes, planes)
        self.bn2 = nn.BatchNorm2d(planes)
        self.downsample = downsample
        self.stride = stride

    def forward(self, x):
        residual = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)

        if self.downsample is not None:
            residual = self.downsample(x)

        out += residual
        out = self.relu(out)

        return out
```

“model.py”

```
class background_resnet(nn.Module):
    def __init__(self, embedding_size, num_classes, backbone='resnet18'):
        super(background_resnet, self).__init__()
        self.backbone = backbone
        # copying modules from pretrained models
        if backbone == 'resnet50':
            self.pretrained = resnet.resnet50(pretrained=False)
        elif backbone == 'resnet101':
            self.pretrained = resnet.resnet101(pretrained=False)
        elif backbone == 'resnet152':
            self.pretrained = resnet.resnet152(pretrained=False)
        elif backbone == 'resnet18':
            self.pretrained = resnet.resnet18(pretrained=False)
        elif backbone == 'resnet34':
            self.pretrained = resnet.resnet34(pretrained=False)
        else:
            raise RuntimeError('unknown backbone: {}'.format(backbone))

        self.fc0 = nn.Linear(128, embedding_size)
        self.bn0 = nn.BatchNorm1d(embedding_size)
        self.relu = nn.ReLU()
        self.last = nn.Linear(embedding_size, num_classes)
```

“model.py”

```
def forward(self, x):
    # input x: minibatch x 1 x 40 x 40
    x = self.pretrained.conv1(x)
    x = self.pretrained.bn1(x)
    x = self.pretrained.relu(x)

    x = self.pretrained.layer1(x)
    x = self.pretrained.layer2(x)
    x = self.pretrained.layer3(x)
    x = self.pretrained.layer4(x)

    out = F.adaptive_avg_pool2d(x,1) # [batch, 128, 1, 1]
    out = torch.squeeze(out) # [batch, n_embed]
    # flatten the out so that the fully connected layer can be connected from here
    out = out.view(x.size(0), -1) # (n_batch, n_embed)
    spk_embedding = self.fc0(out)
    out = F.relu(self.bn0(spk_embedding)) # [batch, n_embed]
    out = self.last(out)

    return spk_embedding, out
```

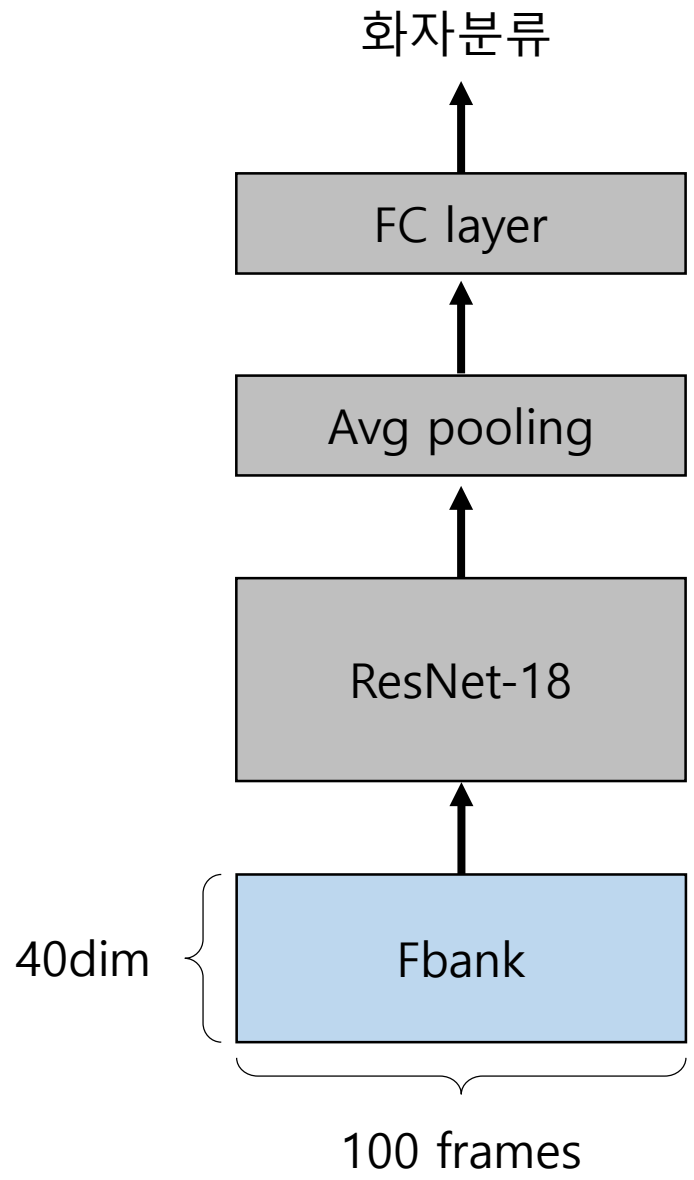
Fbank feature

ResNet-18

average pooling

speaker embedding

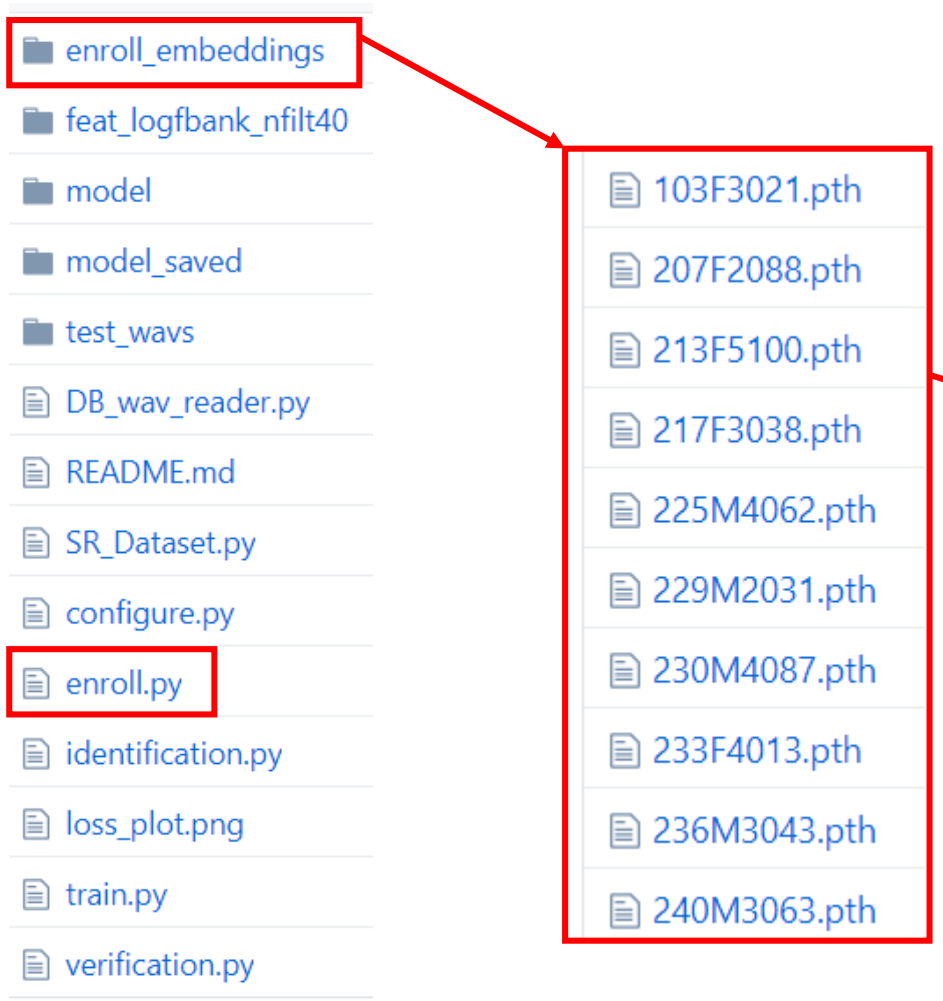
Classifier (FC layer)



폴더 설명


enroll_embeddings
feat_logfbank_nfilt40
model
model_saved
test_wavs
DB_wav_reader.py
README.md
SR_Dataset.py
configure.py
enroll.py
identification.py
loss_plot.png
train.py
verification.py
checkpoint_24.pth
훈련 때 저장되는 모델의 checkpoint
등록 및 테스트 시 이 checkpoint를 불러들임

폴더 설명



"enroll.py"에서 등록 과정을 수행하면,
등록화자 10명의 임베딩 벡터들을 저장

"identification.py" (화자식별) 및
"verification.py" (화자검증)에서 이용

코드 구성

configure.py

- 파일 경로 및 기타 설정 값들 저장

DB_wav_reader.py

- pandas 라이브러리의 dataframe 자료구조를 생성
- 훈련, 등록 및 테스트에서 dataframe을 이용하여 데이터를 불러옴

코드 구성

train.py

- 훈련 데이터를 불러옴 (“DB_wav_reader.py” 이용)
  - 90% 훈련 (train) 데이터 / 10% 검증 (validation) 데이터
- **background model 훈련**

enroll.py

- 등록 데이터를 불러옴 (“DB_wav_reader.py” 이용)
- 훈련된 모델을 이용하여 **등록 화자 (10명)의 임베딩 추출**
- “enroll_embeddings” 폴더에 dictionary 형태로 저장

코드 구성

identification.py

- 저장된 등록 화자들의 임베딩 (10개)을 불러옴
- 테스트 데이터를 불러옴 (“DB_wav_reader.py” 이용)
- 테스트 화자의 임베딩 추출
- **코사인 유사도를 각각 계산하여 가장 유사도가 큰 등록 화자를 출력 (화자식별)**

verification.py

- 저장된 등록 화자의 임베딩 (1개)을 불러옴
- 테스트 데이터를 불러옴 (“DB_wav_reader.py” 이용)
- 테스트 화자의 임베딩 추출
- **두 임베딩 간의 코사인 유사도를 계산 후 threshold 값과 비교하여 검증 수행 (화자검증)**

훈련 (“train.py”)

```
def main():

    # Set hyperparameters
    use_cuda = True # use gpu or cpu
    val_ratio = 10 # Percentage of validation set
    embedding_size = 128
    start = 1 # Start epoch
    n_epochs = 30 # How many epochs?
    end = start + n_epochs # Last epoch

    lr = 1e-1 # Initial learning rate
    wd = 1e-4 # Weight decay (L2 penalty)
    optimizer_type = 'sgd' # ex) sgd, adam, adagrad

    batch_size = 64 # Batch size for training
    valid_batch_size = 16 # Batch size for validation
    use_shuffle = True # Shuffle for training or not

    # Load dataset
    train_dataset, valid_dataset, n_classes = load_dataset(val_ratio)
```

훈련 (“train.py”)

```
# instantiate model and initialize weights
model = background_resnet(embedding_size=embedding_size, num_classes=n_classes)

if use_cuda:
    model.cuda()

# define loss function (criterion), optimizer and scheduler
criterion = nn.CrossEntropyLoss()
optimizer = create_optimizer(optimizer_type, model, lr, wd)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min', patience=2, min_lr=1e-4, verbose=1)

train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                           batch_size=batch_size,
                                           shuffle=use_shuffle)
valid_loader = torch.utils.data.DataLoader(dataset=valid_dataset,
                                           batch_size=valid_batch_size,
                                           shuffle=False,
                                           collate_fn = collate_fn_feat_padded)
```

- learning rate을 조절해주는 scheduler
- 입력으로 validation set의 loss를 넣어줌
- Loss가 plateau 상태면 learning rate을 1/10으로 줄여줌

훈련 (“train.py”)

모델 훈련

validation set의 loss 계산

validation loss를 scheduler의 입력으로 넣어줌

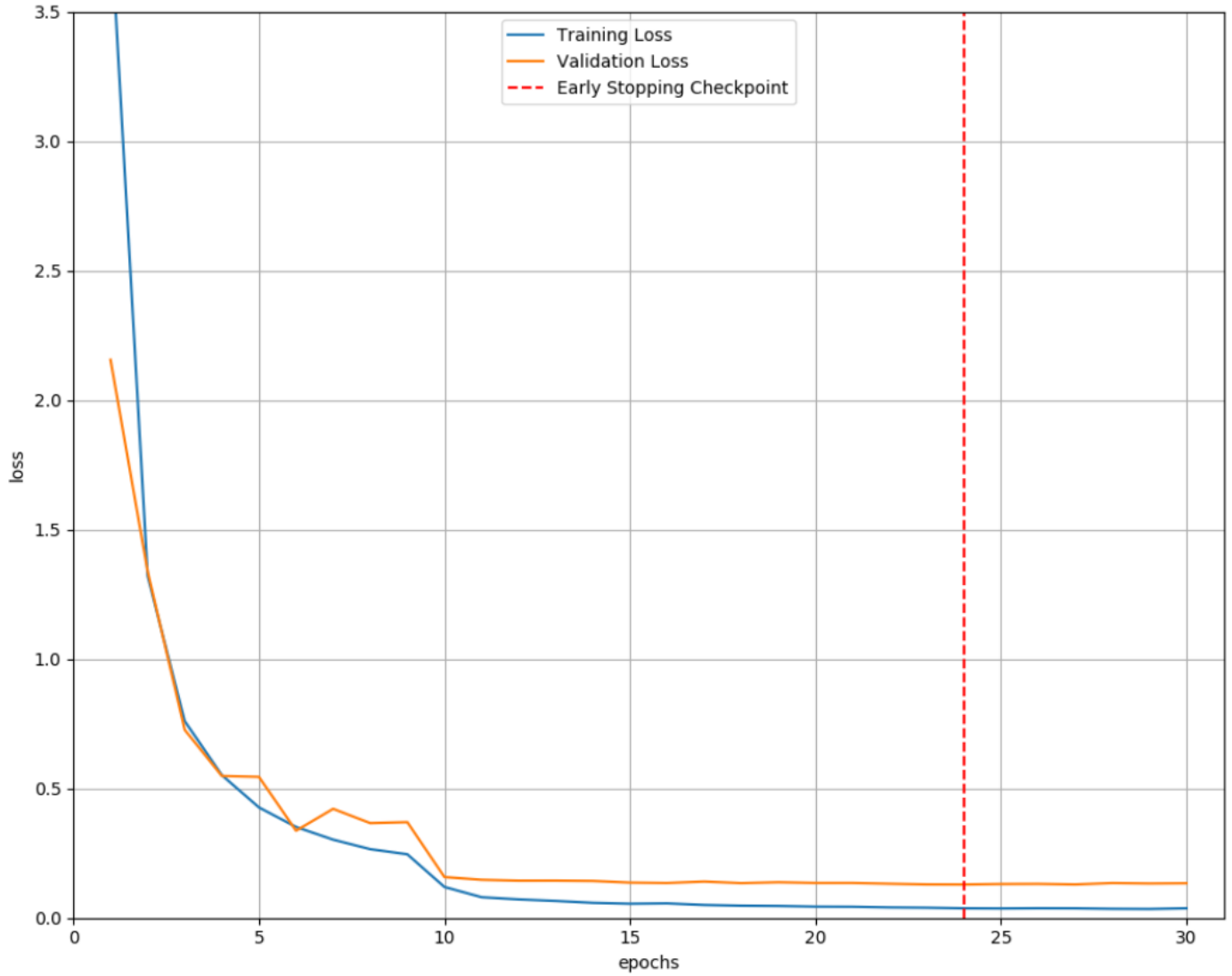
매 epoch마다 훈련된 모델의 checkpoint를 저장함

매 epoch마다 train loss와 valid loss를 저장 후 plot

```
for epoch in range(start, end):

    # train for one epoch
    train_loss = train(train_loader, model, criterion, optimizer, use_cuda, epoch, n_classes)

    # evaluate on validation set
    valid_loss = validate(valid_loader, model, criterion, use_cuda, epoch)

    scheduler.step(valid_loss, epoch)

    # calculate average loss over an epoch
    avg_train_losses.append(train_loss)
    avg_valid_losses.append(valid_loss)

    # do checkpointing
    torch.save({'epoch': epoch + 1, 'state_dict': model.state_dict(),
                'optimizer': optimizer.state_dict()},
               '{}/checkpoint_{}.pth'.format(log_dir, epoch))

# find position of lowest validation loss
minposs = avg_valid_losses.index(min(avg_valid_losses))+1
print('Lowest validation loss at epoch %d' %minposs)

# visualize the loss and learning rate as the network trained
visualize_the_losses(avg_train_losses, avg_valid_losses)
```

훈련 (“train.py”)

“loss_plot.png”



훈련 (“train.py”)

```
(pytorch3) admin@administrator-8:~/Desktop/LG_SR$ python train.py

Training set 21600 utts (90.0%)
Validation set 2400 utts (10.0%)
Total 24000 utts

Number of classes (speakers):
240

Train Epoch:   1 [       0/   21600 (  0%)]     Time 0.134 (0.134)      Loss 5.5429     Acc 0.0000
Train Epoch:   1 [    5376/   21600 ( 25%)]     Time 0.064 (0.070)      Loss 5.1952     Acc 1.0659
Train Epoch:   1 [   10752/   21600 ( 50%)]     Time 0.075 (0.069)      Loss 4.7067     Acc 1.9083
Train Epoch:   1 [   16128/   21600 ( 75%)]     Time 0.084 (0.069)      Loss 4.1807     Acc 3.7277
Train Epoch:   1 [   21504/   21600 ( 99%)]     Time 0.063 (0.068)      Loss 3.7057     Acc 6.3190
  * Validation: Loss 2.1903     Acc 41.9541
Train Epoch:   2 [       0/   21600 (  0%)]     Time 0.037 (0.037)      Loss 1.8758     Acc 50.0000
Train Epoch:   2 [    5376/   21600 ( 25%)]     Time 0.032 (0.031)      Loss 1.6363     Acc 51.0857
Train Epoch:   2 [   10752/   21600 ( 50%)]     Time 0.033 (0.031)      Loss 1.5071     Acc 53.3294
Train Epoch:   2 [   16128/   21600 ( 75%)]     Time 0.031 (0.031)      Loss 1.3909     Acc 55.1330
Train Epoch:   2 [   21504/   21600 ( 99%)]     Time 0.028 (0.031)      Loss 1.2962     Acc 56.7118
  * Validation: Loss 1.1327     Acc 68.6079
```

등록 (“enroll.py”)

```
def main():

    # Settings
    use_cuda = True
    log_dir = 'model_saved'
    embedding_size = 128
    cp_num = 24 # Which checkpoint to use?
    n_classes = 240
    test_frames = 200

    # Load model from checkpoint
    model = load_model(use_cuda, log_dir, cp_num, embedding_size, n_classes)

    # Get the dataframe for enroll DB
    enroll_DB, test_DB = split_enroll_and_test(c.TEST_FEAT_DIR)

    # Where to save embeddings
    embedding_dir = 'enroll_embeddings'

    # Perform the enrollment and save the results
    enroll_per_spk(use_cuda, test_frames, model, enroll_DB, embedding_dir)
```

저장된 checkpoint 중 몇 번째를 불러올 것인가?

“c.TEST_FEAT_DIR”로부터 데이터를 불러온 후, 등록 데이터와 테스트 데이터를 분리함

등록 과정 수행

등록 (“enroll.py”)

pandas 라이브러리의 dataframe 자료구조

- 테이블 형식의 데이터를 다룰 수 있음
- column, row, index의 세 요소로 구성됨
- DB_wav_reader.py에서 정의

```
import pandas as pd
DB = pd.DataFrame()
```

```
>>> enroll_DB
                                              filename speaker_id dataset_id
0  feat_logfbank_nfilt40/test/225M4062/enroll.p   225M4062       test
1  feat_logfbank_nfilt40/test/230M4087/enroll.p   230M4087       test
2  feat_logfbank_nfilt40/test/240M3063/enroll.p   240M3063       test
3  feat_logfbank_nfilt40/test/229M2031/enroll.p   229M2031       test
4  feat_logfbank_nfilt40/test/213F5100/enroll.p   213F5100       test
5  feat_logfbank_nfilt40/test/233F4013/enroll.p   233F4013       test
6  feat_logfbank_nfilt40/test/217F3038/enroll.p   217F3038       test
7  feat_logfbank_nfilt40/test/207F2088/enroll.p   207F2088       test
8  feat_logfbank_nfilt40/test/236M3043/enroll.p   236M3043       test
9  feat_logfbank_nfilt40/test/103F3021/enroll.p   103F3021       test
>>> enroll_DB['filename'][2]
'feat_logfbank_nfilt40/test/240M3063/enroll.p'
>>> enroll_DB['speaker_id'][1]
'230M4087'
>>> enroll_DB['dataset_id'][0]
'test'
```

등록 (“enroll.py”)

“embeddings”라는 dictionary
에 등록 화자의 임베딩을 저장
key : 화자명
value : 임베딩

i번째 파일의 이름과 폴더명(spk)를 구함

i번째 파일을 이용하여 화자임베딩을 추출

“embeddings”에 추출한 임베딩을 저장

```
def enroll_per_spk(use_cuda, test_frames, model, DB, embedding_dir):
    """
    Output the averaged d-vector for each speaker (enrollment)
    Return the dictionary (length of n_spk)
    """
    n_files = len(DB) # 10
    enroll_speaker_list = sorted(set(DB['speaker_id']))

    embeddings = {}

    # Aggregates all the activations
    print("Start to aggregate all the d-vectors per enroll speaker")

    for i in range(n_files):
        filename = DB['filename'][i]
        spk = DB['speaker_id'][i]

        activation = get_embeddings(use_cuda, filename, model, test_frames)
        if spk in embeddings:
            embeddings[spk] += activation
        else:
            embeddings[spk] = activation

        print("Aggregates the activation (spk : %s)" % (spk))
```

등록 (“enroll.py”)

```
(pytorch3) admin@administrator-8:~/Desktop/LG_SR$ python enroll.py
=> loading checkpoint
Start to aggregate all the d-vectors per enroll speaker
Aggregates the activation (spk : 225M4062)
Aggregates the activation (spk : 230M4087)
Aggregates the activation (spk : 240M3063)
Aggregates the activation (spk : 229M2031)
Aggregates the activation (spk : 213F5100)
Aggregates the activation (spk : 233F4013)
Aggregates the activation (spk : 217F3038)
Aggregates the activation (spk : 207F2088)
Aggregates the activation (spk : 236M3043)
Aggregates the activation (spk : 103F3021)
Save the embeddings for 103F3021
Save the embeddings for 207F2088
Save the embeddings for 213F5100
Save the embeddings for 217F3038
Save the embeddings for 225M4062
Save the embeddings for 229M2031
Save the embeddings for 230M4087
Save the embeddings for 233F4013
Save the embeddings for 236M3043
Save the embeddings for 240M3063
```

화자식별 (“identification.py”)

```
def main():

    log_dir = 'model_saved' # Where the checkpoints are saved
    embedding_dir = 'enroll_embeddings' # Where embeddings are saved
    test_dir = 'feat_logfbank_nfilt40/test/' # Where test features are saved

    # Settings
    use_cuda = True # Use cuda or not
    embedding_size = 128 # Dimension of speaker embeddings
    cp_num = 24 # Which checkpoint to use?
    n_classes = 240 # How many speakers in training data?
    test_frames = 100 # Split the test utterance

    # Load model from checkpoint
    model = load_model(use_cuda, log_dir, cp_num, embedding_size, n_classes)

    # Get the dataframe for test DB
    enroll_DB, test_DB = split_enroll_and_test(c.TEST_FEAT_DIR)

    # Load enroll embeddings
    embeddings = load_enroll_embeddings(embedding_dir)
```

“enroll_embeddings” 폴더에 저장된 등록 화자의 임베딩을 불러온다

화자식별 (“identification.py”)

```
spk_list = ['103F3021', '207F2088', '213F5100', '217F3038', '225M4062',\
'229M2031', '230M4087', '233F4013', '236M3043', '240M3063']
```

10명의 등록 화자를 선택하여 입력으로 들어온 테스트 음성이 누구의 음성인지 식별

```
# Set the test speaker
test_speaker = '230M4087'
```

테스트 화자 선택

```
test_path = os.path.join(test_dir, test_speaker, 'test.p')
```

```
# Perform the test
best_spk = perform_identification(use_cuda, model, embeddings, test_path, test_frames, spk_list)
```

화자식별 수행

화자식별 (“identification.py”)

```python
def perform_identification(use_cuda, model, embeddings, test_filename, test_frames, spk_list):
    test_embedding = get_embeddings(use_cuda, test_filename, model, test_frames)
    max_score = -10**8
    best_spk = None
    for spk in spk_list:
        score = F.cosine_similarity(test_embedding, embeddings[spk])
        score = score.data.cpu().numpy()
        if score > max_score:
            max_score = score
            best_spk = spk
    #print("Speaker identification result : %s" %best_spk)
    true_spk = test_filename.split('/')[-2].split('_')[0]
    print("\n=== Speaker identification ===")
    print("True speaker : %s\nPredicted speaker : %s\nResult : %s\n" %(true_spk, best_spk, true_spk==best_spk))
    return best_spk
```

테스트 음성으로부터
화자임베딩 추출

테스트 화자임베딩과 등록 화자임베딩 간의
코사인 유사도 계산

화자식별 (“identification.py”)

```
(pytorch3) admin@administrator-8:~/Desktop/LG_SR$ python identification.py
=> loading checkpoint

=== Speaker identification ===
True speaker : 230M4087
Predicted speaker : 230M4087
Result : True
```

화자검증 (“verification.py”)

```
def main():

    log_dir = 'model_saved' # Where the checkpoints are saved
    embedding_dir = 'enroll_embeddings' # Where embeddings are saved
    test_dir = 'feat_logfbank_nfilt40/test/' # Where test features are saved

    # Settings
    use_cuda = True # Use cuda or not
    embedding_size = 128 # Dimension of speaker embeddings
    cp_num = 24 # Which checkpoint to use?
    n_classes = 240 # How many speakers in training data?
    test_frames = 100 # Split the test utterance

    # Load model from checkpoint
    model = load_model(use_cuda, log_dir, cp_num, embedding_size, n_classes)

    # Get the dataframe for test DB
    enroll_DB, test_DB = split_enroll_and_test(c.TEST_FEAT_DIR)

    # Load enroll embeddings
    embeddings = load_enroll_embeddings(embedding_dir)
```

“enroll_embeddings” 폴더에 저장된 등록 화자의 임베딩을 불러온다

화자검증 (“verification.py”)

```
# Set the true speaker
enroll_speaker = '230M4087'

# Set the claimed speaker
test_speaker = '230M4087'

# Threshold
thres = 0.95

test_path = os.path.join(test_dir, test_speaker, 'test.p')

# Perform the test
perform_verification(use_cuda, model, embeddings, enroll_speaker, test_path, test_frames, thres)
```

등록화자 선택

테스트화자 선택

threshold 설정

화자검증 수행

화자검증 (“verification.py”)

```
def perform_verification(use_cuda, model, embeddings, enroll_speaker, test_filename, test_frames, thres):
    enroll_embedding = embeddings[enroll_speaker]
    test_embedding = get_embeddings(use_cuda, test_filename, model, test_frames)

    score = F.cosine_similarity(test_embedding, enroll_embedding)
    score = score.data.cpu().numpy()

    if score > thres:
        result = 'Accept'
    else:
        result = 'Reject'

    test_spk = test_filename.split('/')[-2].split('_')[0]
    print("\n=== Speaker verification ===")
    print("True speaker: %s\nClaimed speaker : %s\n\nResult : %s\n" %(enroll_speaker, test_spk, result))
    print("Score : %0.4f\nThreshold : %0.2f\n" %(score, thres))
```

등록화자의 임베딩 선택

테스트화자의 임베딩 계산

코사인 유사도 (score) 계산

score와 threshold를 비교

화자검증 (“verification.py”)

```
(pytorch3) admin@administrator-8:~/Desktop/LG_SR$ python verification.py
=> loading checkpoint

=== Speaker verification ===
True speaker: 230M4087
Claimed speaker : 230M4087

Result : Accept

Score : 0.9556
Threshold : 0.95
```

```
(pytorch3) admin@administrator-8:~/Desktop/LG_SR$ python verification.py
=> loading checkpoint

=== Speaker verification ===
True speaker: 230M4087
Claimed speaker : 207F2088

Result : Reject

Score : 0.8026
Threshold : 0.95
```

개별 실습

- **순서**

① train.py
② enroll.py
③ identification.py
④ verification.py

- **실습사항**

1. 순서에 맞춰 직접 코드 실행해보기
2. hyperparameter 바꿔가며 훈련해보기
   - Loss function 변화 확인
   - 테스트 결과 확인
3. 목소리가 비슷한 화자일수록 코사인 유사도가 높은지 확인
4. threshold 값에 따른 화자검증 결과 비교

- **심화학습**

1. voxceleb DB를 적용
   http://www.robots.ox.ac.uk/~vgg/data/voxceleb/
2. 다른 loss function 적용
   ex) center loss, angular softmax loss,…
3. 다른 모델 적용
   ex) VGGNet, LSTM,…