实验七、声纹识别

一、实验目的

1.掌握声纹识别的基本原理。
2.掌握声模型纹识别的实现方法。

二、实验要求

基于TensorFlow框架及Python相关模块搭建声纹识别模型并训练，最后实现对未知人的声音进行识别。

三、实验原理

声纹识别中的说话人识别主要分为文本依赖说话人鉴别（TD-SV）和文本独立说话者鉴别（TI-SV），该实验以《GENERALIZED END-TO-END LOSS FOR SPEAKER VERIFICATION》（GE2E）文章为原理进行实验，在无噪声的语音数据下使用端到端的方法实现TI-SV。主要原理如下：

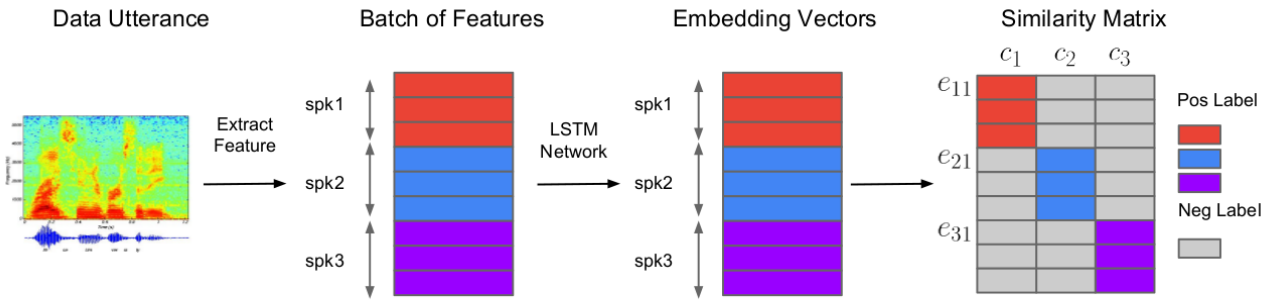

Fig. 1. System overview. Different colors indicate utterances/embeddings from different speakers.

基于批处理的训练方法，使同一批中每个说话者与其最相似的说话者声纹特征变的不同。

使用泛化端到端（GE2E）损失函数。提升声纹识别模型的训练效率。GE2E 损失函数在训练过程中，依据每一步所遇到的最困难样本来更新网络。此外，GE2E 也不需要额外的样本选择步骤。使用该损失函数的模型能学到更好的模型，极大降低错误率（EER）指标和训练的时间。

模型的评价指标为Equal Error Rate (EER)，是说话人识别和确认的常用评价指标。是一种使错误接受率（FAR） 和错误拒绝率（FRR）的一个相对平衡点阈值点，然后这个阈值点可以作为实际使用阶段的固定的阈值。

四、实验所用工具及数据集

1.主要工具
TensorFlow-gpu-1.10.0、librosa等
2.数据集
该CSTR VCTK语料库包括由109名母语为英语且具有各种口音的语音数据。每位发言者都会读出大约400个句子，其中大部分是从英文报纸选出。语料库下载地址：https://datashare.is.ed.ac.uk/handle/10283/2651。

实际使用其中90%作为模型训练数据，10%作为测试数据。可根据实际情况调整划分比例。

五、实验步骤与方法

一、数据预处理

按一定比例划分训练集与数据集并采用滑动窗口方法利用librosa语音处理库对语音的MFCC（梅尔频率倒谱系数）特征进行提取，并保存为.npy格式的文件至指定文件夹，文件表示每一位说话者每一条口语的声纹特征。

```
import numpy as np
import os
import librosa
import tensorflow as  tf
import time
from tensorflow.contrib import rnn
import random
```

定义全局配置参数。

```
class CONFIG(object):
    def __init__(self):
        self.sr = 8000
        self.nfft = 512
        self.window = 0.025
        self.hop = 0.01
        self.tisv_frame = 180
        self.train_path = "./train_tisv"
        self.test_path = "./test_tisv"

        self.hidden = 128
        self.proj = 64
        self.num_layer = 3
        self.restore = False
        self.model_path = "./model"
        self.model_num = 5

        self.train = False
        self.N = 8
        self.M = 10
        self.loss = "softmax"
        self.optim = "sgd"
        self.lr = 1e-2
        self.beta1 = 0.5
        self.beta2 = 0.9
        self.iteration = 60000
```

```
config = CONFIG()
audio_path = "./wav48"
```

```
def save_spectrogram_tisv():
    print("start text independent utterance feature extraction")
    os.makedirs(config.train_path, exist_ok=True)   # make folder to save train file
```

```
    os.makedirs(config.test_path, exist_ok=True)    # make folder to save test file

    utter_min_len = (config.tisv_frame * config.hop + config.window) * config.sr    # lower bound of utterance length
    total_speaker_num = len(os.listdir(audio_path))
    train_speaker_num= (total_speaker_num//10)*9            # split total data 90% train and 10% test
    print("total speaker number : %d"%total_speaker_num)
    print("train : %d, test : %d"%(train_speaker_num, total_speaker_num-train_speaker_num))
    for i, folder in enumerate(os.listdir(audio_path)):
        if i <= 107:
            continue
        speaker_path = os.path.join(audio_path, folder)     # path of each speaker
        print("%dth speaker processing..."%i)
        utterances_spec = []
        for utter_name in os.listdir(speaker_path):
            utter_path = os.path.join(speaker_path, utter_name)         # path of each utterance
            utter, sr = librosa.core.load(utter_path, config.sr)        # load utterance audio
            intervals = librosa.effects.split(utter, top_db=20)         # voice activity detection
            for interval in intervals:
                if (interval[1]-interval[0]) > utter_min_len:           # If partial utterance is sufficient long,
                    utter_part = utter[interval[0]:interval[1]]         # save first and last 180 frames of spectrogram.
                    S = librosa.core.stft(y=utter_part, n_fft=config.nfft,
                                          win_length=int(config.window * sr), hop_length=int(config.hop * sr))
                    S = np.abs(S) ** 2
                    mel_basis = librosa.filters.mel(sr=config.sr, n_fft=config.nfft, n_mels=40)
                    S = np.log10(np.dot(mel_basis, S) + 1e-6)           # log mel spectrogram of utterances

                    utterances_spec.append(S[:, :config.tisv_frame])    # first 180 frames of partial utterance
                    utterances_spec.append(S[:, -config.tisv_frame:])   # last 180 frames of partial utterance

        utterances_spec = np.array(utterances_spec)
        print(utterances_spec.shape)
        if i<train_speaker_num:      # save spectrogram as numpy file
            np.save(os.path.join(config.train_path, "speaker%d.npy"%i), utterances_spec)
        else:
            np.save(os.path.join(config.test_path, "speaker%d.npy"%(i-train_speaker_num)), utterances_spec)

```

```
save_spectrogram_tisv()
```

```
start text independent utterance feature extraction
total speaker number : 109
train : 90, test : 19
108th speaker processing...
(260, 40, 180)
```

二、构建训练模型

1、在步骤一中生成的.npy文件中，随机选择N位说话者并对应选择M条口语声纹特征，组成的batch大小N×M。

```
def random_batch(speaker_num=config.N, utter_num=config.M, shuffle=True, noise_filenum=None, utter_start=0):
    # data path
    if config.train:
        path = config.train_path
    else:
        path = config.test_path

    np_file_list = os.listdir(path)
    total_speaker = len(np_file_list)

    if shuffle:
        selected_files = random.sample(np_file_list, speaker_num)  # select random N speakers
    else:
        selected_files = np_file_list[:speaker_num]                # select first N speakers

    utter_batch = []
    for file in selected_files:
        utters = np.load(os.path.join(path, file))        # load utterance spectrogram of selected speaker
        if shuffle:
            utter_index = np.random.randint(0, utters.shape[0], utter_num)   # select M utterances per speaker
            utter_batch.append(utters[utter_index])       # each speakers utterance [M, n_mels, frames] is appended
        else:
            utter_batch.append(utters[utter_start: utter_start+utter_num])

    utter_batch = np.concatenate(utter_batch, axis=0)     # utterance batch [batch(NM), n_mels, frames]

    if config.train:
        frame_slice = np.random.randint(140,181)          # for train session, random slicing of input batch
        utter_batch = utter_batch[:,:,:frame_slice]
    else:
```

```
        utter_batch = utter_batch[:,:,:160]               # for train session, fixed length slicing of input batch

    utter_batch = np.transpose(utter_batch, axes=(2,0,1))     # transpose [frames, batch, n_mels]

    return utter_batch
```

2、定义数据归一化函数和相似度函数

```
def normalize(x):
    """ normalize the last dimension vector of the input matrix
    :return: normalized input
    """
    return x/tf.sqrt(tf.reduce_sum(x**2, axis=-1, keepdims=True)+1e-6)


def cossim(x,y, normalized=True):
    """ calculate similarity between tensors
    :return: cos similarity tf op node
    """
    if normalized:
        return tf.reduce_sum(x*y)
    else:
        x_norm = tf.sqrt(tf.reduce_sum(x**2)+1e-6)
        y_norm = tf.sqrt(tf.reduce_sum(y**2)+1e-6)
        return tf.reduce_sum(x*y)/x_norm/y_norm
```

3、计算相似度矩阵

$$\mathbf{S}_{ji,k} = w \cdot \cos(\mathbf{e}_{ji}, \mathbf{c}_k) + b$$

```
def similarity(embedded, w, b, N=config.N, M=config.M, P=config.proj, center=None):
    """ Calculate similarity matrix from embedded utterance batch (NM x embed_dim) eq.(9)
        Input center to test enrollment. (embedded for verification)
    :return: tf similarity matrix (NM x N)
    """
    embedded_split = tf.reshape(embedded, shape=[N, M, P])

    if center is None:
        center = normalize(tf.reduce_mean(embedded_split, axis=1))              # [N,P] normalized center vectors eq.(1)
        center_except = normalize(tf.reshape(tf.reduce_sum(embedded_split, axis=1, keepdims=True)
                                             - embedded_split, shape=[N*M,P]))  # [NM,P] center vectors eq.(8)
        # make similarity matrix eq.(9)
        S = tf.concat(
```

```
            [tf.concat([tf.reduce_sum(center_except[i*M:(i+1)*M,:]*embedded_split[j,:,:], axis=1, keepdims=True) if i==j
                        else tf.reduce_sum(center[i:(i+1),:]*embedded_split[j,:,:], axis=1, keepdims=True) for i in range(N)],
                       axis=1) for j in range(N)], axis=0)
    else :
        # If center(enrollment) exist, use it.
        S = tf.concat(
            [tf.concat([tf.reduce_sum(center[i:(i + 1), :] * embedded_split[j, :, :], axis=1, keepdims=True) for i
                        in range(N)],
                       axis=1) for j in range(N)], axis=0)

    S = tf.abs(w)*S+b   # rescaling

    return S
```

4、根据相似度矩阵计算损失

```
def loss_cal(S, type="softmax", N=config.N, M=config.M):
    """ calculate loss with similarity matrix(S) eq.(6) (7)
    :type: "softmax" or "contrast"
    :return: loss
    """
    S_correct = tf.concat([S[i*M:(i+1)*M, i:(i+1)] for i in range(N)], axis=0)  # colored entries in Fig.1

    if type == "softmax":
        total = -tf.reduce_sum(S_correct-tf.log(tf.reduce_sum(tf.exp(S), axis=1, keepdims=True) + 1e-6))
    elif type == "contrast":
        S_sig = tf.sigmoid(S)
        S_sig = tf.concat([tf.concat([0*S_sig[i*M:(i+1)*M, j:(j+1)] if i==j
                                      else S_sig[i*M:(i+1)*M, j:(j+1)] for j in range(N)], axis=1)
                           for i in range(N)], axis=0)
        total = tf.reduce_sum(1-tf.sigmoid(S_correct)+tf.reduce_max(S_sig, axis=1, keepdims=True))
    else:
        raise AssertionError("loss type should be softmax or contrast !")

    return total
```

5、定义模型优化器函数

```
def optim(lr):
    """ return optimizer determined by configuration
    :return: tf optimizer
    """
    if config.optim == "sgd":
        return tf.train.GradientDescentOptimizer(lr)
    elif config.optim == "rmsprop":
        return tf.train.RMSPropOptimizer(lr)
    elif config.optim == "adam":
        return tf.train.AdamOptimizer(lr, beta1=config.beta1, beta2=config.beta2)
    else:
        raise AssertionError("Wrong optimizer type!")
```

6、定义训练模型函数

本实验使用了3层LSTM网络，其输出作为Embedding d-Vector，再进行L2正则化，得到的向量就是说话人的声纹表征。

```
def train(path):
    tf.reset_default_graph()    # reset graph

    # draw graph
    batch = tf.placeholder(shape= [None, config.N*config.M, 40], dtype=tf.float32)  # input batch (time x batch x n_mel)
    lr = tf.placeholder(dtype= tf.float32)  # learning rate
    global_step = tf.Variable(0, name='global_step', trainable=False)
    w = tf.get_variable("w", initializer= np.array([10], dtype=np.float32))
    b = tf.get_variable("b", initializer= np.array([-5], dtype=np.float32))

    # embedding lstm (3-layer default)
    with tf.variable_scope("lstm"):
        lstm_cells = [tf.contrib.rnn.LSTMCell(num_units=config.hidden, num_proj=config.proj) for i in range(config.num_layer)]
        lstm = tf.contrib.rnn.MultiRNNCell(lstm_cells)    # define lstm op and variables
        outputs, _ = tf.nn.dynamic_rnn(cell=lstm, inputs=batch, dtype=tf.float32, time_major=True)   # for TI-VS must use dynamic rnn
        embedded = outputs[-1]                            # the last ouput is the embedded d-vector
        embedded = normalize(embedded)                    # normalize
    print("embedded size: ", embedded.shape)

    # loss
    sim_matrix = similarity(embedded, w, b)
    print("similarity matrix size: ", sim_matrix.shape)
    loss = loss_cal(sim_matrix, type=config.loss)

    # optimizer operation
    trainable_vars= tf.trainable_variables()                # get variable list
    optimizer= optim(lr)                                    # get optimizer (type is determined by configuration)
```

```
    grads, vars= zip(*optimizer.compute_gradients(loss))    # compute gradients of variables with respect to loss
    grads_clip, _ = tf.clip_by_global_norm(grads, 3.0)      # l2 norm clipping by 3
    grads_rescale= [0.01*grad for grad in grads_clip[:2]] + grads_clip[2:]   # smaller gradient scale for w, b
    train_op= optimizer.apply_gradients(zip(grads_rescale, vars), global_step= global_step)   # gradient update operation

    # check variables memory
    variable_count = np.sum(np.array([np.prod(np.array(v.get_shape().as_list())) for v in trainable_vars]))
    print("total variables :", variable_count)

    # record loss
    loss_summary = tf.summary.scalar("loss", loss)
    merged = tf.summary.merge_all()
    saver = tf.train.Saver()

    # training session
    with tf.Session() as sess:
        tf.global_variables_initializer().run()
        os.makedirs(os.path.join(path, "Check_Point"), exist_ok=True)  # make folder to save model
        os.makedirs(os.path.join(path, "logs"), exist_ok=True)          # make folder to save log
        writer = tf.summary.FileWriter(os.path.join(path, "logs"), sess.graph)
        epoch = 0
        lr_factor = 1   # lr decay factor ( 1/2 per 10000 iteration)
        loss_acc = 0    # accumulated loss ( for running average of loss)

        for iter in range(config.iteration):
            # run forward and backward propagation and update parameters
            _, loss_cur, summary = sess.run([train_op, loss, merged],
                                    feed_dict={batch: random_batch(), lr: config.lr*lr_factor})

            loss_acc += loss_cur    # accumulated loss for each 100 iteration

            if iter % 10 == 0:
                writer.add_summary(summary, iter)   # write at tensorboard
            if (iter+1) % 100 == 0:
                print("(iter : %d) loss: %.4f" % ((iter+1),loss_acc/100))
                loss_acc = 0                        # reset accumulated loss
            if (iter+1) % 10000 == 0:
                lr_factor /= 2                      # lr decay
                print("learning rate is decayed! current lr : ", config.lr*lr_factor)
            if (iter+1) % 10000 == 0:
                saver.save(sess, os.path.join(path, "./Check_Point/model.ckpt"), global_step=iter//10000)
                print("model is saved!")
```

7、定义测试模型函数

利用已处理好的测试数据进行模型的测试使用，并根据评价指标EER分析实验结果。

```
# Test Session
def test(path):
    tf.reset_default_graph()

    # draw graph
    enroll = tf.placeholder(shape=[None, config.N*config.M, 40], dtype=tf.float32) # enrollment batch (time x batch x n_mel)
    verif = tf.placeholder(shape=[None, config.N*config.M, 40], dtype=tf.float32)  # verification batch (time x batch x n_mel)
    batch = tf.concat([enroll, verif], axis=1)

    # embedding lstm (3-layer default)
    with tf.variable_scope("lstm"):
        lstm_cells = [tf.contrib.rnn.LSTMCell(num_units=config.hidden, num_proj=config.proj) for i in range(config.num_layer)]
        lstm = tf.contrib.rnn.MultiRNNCell(lstm_cells)    # make lstm op and variables
        outputs, _ = tf.nn.dynamic_rnn(cell=lstm, inputs=batch, dtype=tf.float32, time_major=True)   # for TI-VS must use dynamic rnn
        embedded = outputs[-1]                            # the last ouput is the embedded d-vector
        embedded = normalize(embedded)                    # normalize

    print("embedded size: ", embedded.shape)

    # enrollment embedded vectors (speaker model)
    enroll_embed = normalize(tf.reduce_mean(tf.reshape(embedded[:config.N*config.M, :], shape= [config.N, config.M, -1]), axis=1))
    # verification embedded vectors
    verif_embed = embedded[config.N*config.M:, :]

    similarity_matrix = similarity(embedded=verif_embed, w=1., b=0., center=enroll_embed)

    saver = tf.train.Saver(var_list=tf.global_variables())
    with tf.Session() as sess:
        tf.global_variables_initializer().run()

        # load model
        print("model path :", path)
        ckpt = tf.train.get_checkpoint_state(checkpoint_dir=os.path.join(path, "Check_Point"))
        ckpt_list = ckpt.all_model_checkpoint_paths
        loaded = 0
        for model in ckpt_list:
            if config.model_num == int(model[-1]):    # find ckpt file which matches configuration model number
                print("ckpt file is loaded !", model)
                loaded = 1
                saver.restore(sess, model)  # restore variables from selected ckpt file
                break
```

```
        if loaded == 0:
            raise AssertionError("ckpt file does not exist! Check config.model_num or config.model_path.")

        print("test file path : ", config.test_path)

        # return similarity matrix after enrollment and verification
        time1 = time.time() # for check inference time
        S = sess.run(similarity_matrix, feed_dict={enroll:random_batch(shuffle=False), verif:random_batch(shuffle=False, utter_start=config.M)})
        S = S.reshape([config.N, config.M, -1])
        time2 = time.time()

        np.set_printoptions(precision=2)
        print("inference time for %d utterences : %0.2fs"%(2*config.M*config.N, time2-time1))
        print(S)    # print similarity matrix

        # calculating EER
        diff = 1; EER=0; EER_thres = 0; EER_FAR=0; EER_FRR=0

        # through thresholds calculate false acceptance ratio (FAR) and false reject ratio (FRR)
        for thres in [0.01*i+0.5 for i in range(50)]:
            S_thres = S>thres

            # False acceptance ratio = false acceptance / mismatched population (enroll speaker != verification speaker)
            FAR = sum([np.sum(S_thres[i])-np.sum(S_thres[i,:,i]) for i in range(config.N)])/(config.N-1)/config.M/config.N

            # False reject ratio = false reject / matched population (enroll speaker = verification speaker)
            FRR = sum([config.M-np.sum(S_thres[i][:,i]) for i in range(config.N)])/config.M/config.N

            # Save threshold when FAR = FRR (=EER)
            if diff> abs(FAR-FRR):
                diff = abs(FAR-FRR)
                EER = (FAR+FRR)/2
                EER_thres = thres
                EER_FAR = FAR
                EER_FRR = FRR

        print("\nEER : %0.2f (thres:%0.2f, FAR:%0.2f, FRR:%0.2f)"%(EER,EER_thres,EER_FAR,EER_FRR))
```

```
config.train= True
train("./model")
```

```
embedded size:  (80, 64)
similarity matrix size:  (80, 8)
total variables : 210434
(iter : 100) loss: 125.1124
(iter : 200) loss: 102.2868
(iter : 300) loss: 89.9621
(iter : 400) loss: 86.9429
(iter : 500) loss: 79.9499
(iter : 600) loss: 75.2657
(iter : 700) loss: 70.0881
(iter : 800) loss: 65.6341
(iter : 900) loss: 62.9165
(iter : 1000) loss: 61.5927
(iter : 1100) loss: 54.8214
(iter : 1200) loss: 51.2984
(iter : 1300) loss: 48.3105
(iter : 1400) loss: 45.9122
(iter : 1500) loss: 41.6896
(iter : 1600) loss: 43.2339
(iter : 1700) loss: 38.5662
(iter : 1800) loss: 36.0750
(iter : 1900) loss: 35.9866
(iter : 2000) loss: 35.6972
(iter : 2100) loss: 32.1313
(iter : 2200) loss: 32.2583
(iter : 2300) loss: 31.1465
(iter : 2400) loss: 28.6453
(iter : 2500) loss: 29.0325
(iter : 2600) loss: 28.3069
(iter : 2700) loss: 27.3037
(iter : 2800) loss: 26.8961
(iter : 2900) loss: 26.5261
(iter : 3000) loss: 27.4743
(iter : 3100) loss: 26.0053
(iter : 3200) loss: 25.8477
(iter : 3300) loss: 24.0968
(iter : 3400) loss: 25.1196
(iter : 3500) loss: 23.1311
(iter : 3600) loss: 23.4995
(iter : 3700) loss: 23.4513
(iter : 3800) loss: 21.4265
(iter : 3900) loss: 23.1132
(iter : 4000) loss: 22.9096
(iter : 4100) loss: 20.7972
(iter : 4200) loss: 20.9216
(iter : 4300) loss: 22.6000
(iter : 4400) loss: 21.3948
(iter : 4500) loss: 20.1084
(iter : 4600) loss: 19.0182
(iter : 4700) loss: 20.5251
(iter : 4800) loss: 20.6271
(iter : 4900) loss: 20.0620
(iter : 5000) loss: 18.2316
```

```
(iter : 5100) loss: 19.5083
(iter : 5200) loss: 17.8354
(iter : 5300) loss: 19.2086
(iter : 5400) loss: 18.4276
(iter : 5500) loss: 18.0853
(iter : 5600) loss: 16.8695
(iter : 5700) loss: 16.1810
(iter : 5800) loss: 17.3694
(iter : 5900) loss: 17.9695
(iter : 6000) loss: 18.7489
(iter : 6100) loss: 14.8636
(iter : 6200) loss: 16.2349
(iter : 6300) loss: 17.2699
(iter : 6400) loss: 15.9706
(iter : 6500) loss: 17.6411
(iter : 6600) loss: 15.8970
(iter : 6700) loss: 16.7262
(iter : 6800) loss: 16.3059
(iter : 6900) loss: 16.4714
(iter : 7000) loss: 15.3530
(iter : 7100) loss: 14.8779
(iter : 7200) loss: 15.7628
(iter : 7300) loss: 15.7174
(iter : 7400) loss: 15.3888
(iter : 7500) loss: 15.8906
(iter : 7600) loss: 14.1748
(iter : 7700) loss: 14.7313
(iter : 7800) loss: 15.0035
(iter : 7900) loss: 15.3389
(iter : 8000) loss: 14.7931
(iter : 8100) loss: 14.1820
(iter : 8200) loss: 14.6710
(iter : 8300) loss: 14.4680
(iter : 8400) loss: 12.9227
(iter : 8500) loss: 14.4477
(iter : 8600) loss: 14.0131
(iter : 8700) loss: 14.1133
(iter : 8800) loss: 12.9073
(iter : 8900) loss: 14.6619
(iter : 9000) loss: 13.1309
(iter : 9100) loss: 12.8601
(iter : 9200) loss: 13.0869
(iter : 9300) loss: 13.3302
(iter : 9400) loss: 12.3137
(iter : 9500) loss: 13.0615
(iter : 9600) loss: 13.2034
(iter : 9700) loss: 13.0240
(iter : 9800) loss: 12.0275
(iter : 9900) loss: 12.6227
(iter : 10000) loss: 13.8931
learning rate is decayed! current lr :  0.005
model is saved!
(iter : 10100) loss: 11.3030
```

```
(iter : 10200) loss: 10.1317
(iter : 10300) loss: 11.6815
(iter : 10400) loss: 10.6280
(iter : 10500) loss: 11.5178
(iter : 10600) loss: 10.2910
(iter : 10700) loss: 10.0618
(iter : 10800) loss: 10.4116
(iter : 10900) loss: 11.0984
(iter : 11000) loss: 10.8993
(iter : 11100) loss: 10.2046
(iter : 11200) loss: 10.3189
(iter : 11300) loss: 10.6825
(iter : 11400) loss: 10.2830
(iter : 11500) loss: 10.5885
(iter : 11600) loss: 9.9892
(iter : 11700) loss: 10.4700
(iter : 11800) loss: 9.3798
(iter : 11900) loss: 9.9150
(iter : 12000) loss: 10.9176
(iter : 12100) loss: 10.1597
(iter : 12200) loss: 9.9589
(iter : 12300) loss: 9.4352
(iter : 12400) loss: 10.8907
(iter : 12500) loss: 10.8410
(iter : 12600) loss: 10.9076
(iter : 12700) loss: 9.6158
(iter : 12800) loss: 9.6700
(iter : 12900) loss: 9.3035
(iter : 13000) loss: 9.9638
(iter : 13100) loss: 8.7488
(iter : 13200) loss: 9.7100
(iter : 13300) loss: 10.0202
(iter : 13400) loss: 9.0262
(iter : 13500) loss: 10.2791
(iter : 13600) loss: 9.8496
(iter : 13700) loss: 9.5234
(iter : 13800) loss: 10.0217
(iter : 13900) loss: 9.4878
(iter : 14000) loss: 9.5881
(iter : 14100) loss: 10.0928
(iter : 14200) loss: 8.4435
(iter : 14300) loss: 9.4998
(iter : 14400) loss: 9.3261
(iter : 14500) loss: 9.4715
(iter : 14600) loss: 9.6004
(iter : 14700) loss: 9.3574
(iter : 14800) loss: 9.0400
(iter : 14900) loss: 9.4266
(iter : 15000) loss: 9.6147
(iter : 15100) loss: 8.1049
(iter : 15200) loss: 10.1856
(iter : 15300) loss: 9.2970
(iter : 15400) loss: 9.4341
```

```
(iter : 15500) loss: 9.9012
(iter : 15600) loss: 8.7888
(iter : 15700) loss: 8.5273
(iter : 15800) loss: 9.0859
(iter : 15900) loss: 8.5088
(iter : 16000) loss: 8.1488
(iter : 16100) loss: 9.0367
(iter : 16200) loss: 8.8542
(iter : 16300) loss: 8.4030
(iter : 16400) loss: 9.2126
(iter : 16500) loss: 8.8424
(iter : 16600) loss: 8.6824
(iter : 16700) loss: 8.3212
(iter : 16800) loss: 9.3472
(iter : 16900) loss: 8.7888
(iter : 17000) loss: 8.5208
(iter : 17100) loss: 8.3289
(iter : 17200) loss: 9.2287
(iter : 17300) loss: 8.5320
(iter : 17400) loss: 8.5493
(iter : 17500) loss: 8.8745
(iter : 17600) loss: 9.2978
(iter : 17700) loss: 8.4141
(iter : 17800) loss: 9.4525
(iter : 17900) loss: 9.1388
(iter : 18000) loss: 9.3942
(iter : 18100) loss: 9.0028
(iter : 18200) loss: 8.0412
(iter : 18300) loss: 8.2772
(iter : 18400) loss: 7.6248
(iter : 18500) loss: 7.7044
(iter : 18600) loss: 7.8588
(iter : 18700) loss: 8.2669
(iter : 18800) loss: 7.7532
(iter : 18900) loss: 7.6978
(iter : 19000) loss: 8.0033
(iter : 19100) loss: 7.9617
(iter : 19200) loss: 8.1638
(iter : 19300) loss: 7.7100
(iter : 19400) loss: 8.4997
(iter : 19500) loss: 8.7061
(iter : 19600) loss: 8.2648
(iter : 19700) loss: 8.5872
(iter : 19800) loss: 7.9430
(iter : 19900) loss: 8.0119
(iter : 20000) loss: 7.6592
learning rate is decayed! current lr :  0.0025
model is saved!
(iter : 20100) loss: 6.8616
(iter : 20200) loss: 6.9169
(iter : 20300) loss: 7.8132
(iter : 20400) loss: 7.8254
(iter : 20500) loss: 8.3232
```

```
(iter : 20600) loss: 7.5797
(iter : 20700) loss: 8.0273
(iter : 20800) loss: 7.6046
(iter : 20900) loss: 7.9540
(iter : 21000) loss: 6.8477
(iter : 21100) loss: 8.0044
(iter : 21200) loss: 7.5676
(iter : 21300) loss: 6.9838
(iter : 21400) loss: 6.5275
(iter : 21500) loss: 7.9485
(iter : 21600) loss: 7.6059
(iter : 21700) loss: 6.6720
(iter : 21800) loss: 7.0466
(iter : 21900) loss: 6.2690
(iter : 22000) loss: 7.7404
(iter : 22100) loss: 6.5648
(iter : 22200) loss: 7.0133
(iter : 22300) loss: 6.2991
(iter : 22400) loss: 6.8944
(iter : 22500) loss: 6.3249
(iter : 22600) loss: 6.8914
(iter : 22700) loss: 7.6698
(iter : 22800) loss: 7.3043
(iter : 22900) loss: 6.7923
(iter : 23000) loss: 7.0147
(iter : 23100) loss: 7.2050
(iter : 23200) loss: 7.9540
(iter : 23300) loss: 7.0974
(iter : 23400) loss: 7.0630
(iter : 23500) loss: 7.2091
(iter : 23600) loss: 6.3066
(iter : 23700) loss: 7.4328
(iter : 23800) loss: 6.1453
(iter : 23900) loss: 7.0220
(iter : 24000) loss: 6.9030
(iter : 24100) loss: 6.6377
(iter : 24200) loss: 7.4818
(iter : 24300) loss: 7.2563
(iter : 24400) loss: 6.8584
(iter : 24500) loss: 6.4224
(iter : 24600) loss: 7.2655
(iter : 24700) loss: 6.3695
(iter : 24800) loss: 6.5579
(iter : 24900) loss: 6.4468
(iter : 25000) loss: 6.4815
(iter : 25100) loss: 6.7222
(iter : 25200) loss: 6.3811
(iter : 25300) loss: 6.1798
(iter : 25400) loss: 6.6120
(iter : 25500) loss: 6.6860
(iter : 25600) loss: 7.2635
(iter : 25700) loss: 6.8251
(iter : 25800) loss: 6.5754
```

```
(iter : 25900) loss: 6.8174
(iter : 26000) loss: 7.7046
(iter : 26100) loss: 7.2207
(iter : 26200) loss: 7.0398
(iter : 26300) loss: 6.5856
(iter : 26400) loss: 6.7113
(iter : 26500) loss: 6.9175
(iter : 26600) loss: 6.8487
(iter : 26700) loss: 6.1153
(iter : 26800) loss: 7.0726
(iter : 26900) loss: 6.4806
(iter : 27000) loss: 7.1133
(iter : 27100) loss: 6.1659
(iter : 27200) loss: 7.2581
(iter : 27300) loss: 6.2285
(iter : 27400) loss: 7.5509
(iter : 27500) loss: 7.0549
(iter : 27600) loss: 5.8054
(iter : 27700) loss: 6.9110
(iter : 27800) loss: 6.9032
(iter : 27900) loss: 7.0663
(iter : 28000) loss: 6.7402
(iter : 28100) loss: 6.7317
(iter : 28200) loss: 6.8238
(iter : 28300) loss: 6.0625
(iter : 28400) loss: 6.4912
(iter : 28500) loss: 6.2927
(iter : 28600) loss: 6.0508
(iter : 28700) loss: 5.8554
(iter : 28800) loss: 6.1459
(iter : 28900) loss: 6.6405
(iter : 29000) loss: 6.1738
(iter : 29100) loss: 6.7811
(iter : 29200) loss: 6.3596
(iter : 29300) loss: 6.6660
(iter : 29400) loss: 7.1217
(iter : 29500) loss: 7.2853
(iter : 29600) loss: 6.0356
(iter : 29700) loss: 6.9032
(iter : 29800) loss: 6.5223
(iter : 29900) loss: 6.9550
(iter : 30000) loss: 6.4661
learning rate is decayed! current lr :  0.00125
model is saved!
(iter : 30100) loss: 6.4794
(iter : 30200) loss: 6.0957
(iter : 30300) loss: 5.8880
(iter : 30400) loss: 6.8376
(iter : 30500) loss: 5.5698
(iter : 30600) loss: 6.2785
(iter : 30700) loss: 5.2932
(iter : 30800) loss: 5.7432
(iter : 30900) loss: 5.4160
```

```
(iter : 31000) loss: 6.5425
(iter : 31100) loss: 5.7405
(iter : 31200) loss: 6.9640
(iter : 31300) loss: 5.5323
(iter : 31400) loss: 6.5483
(iter : 31500) loss: 5.9443
(iter : 31600) loss: 5.8195
(iter : 31700) loss: 5.6073
(iter : 31800) loss: 6.0223
(iter : 31900) loss: 5.5022
(iter : 32000) loss: 5.6353
(iter : 32100) loss: 5.6601
(iter : 32200) loss: 6.0250
(iter : 32300) loss: 6.8285
(iter : 32400) loss: 5.9613
(iter : 32500) loss: 5.6028
(iter : 32600) loss: 5.9611
(iter : 32700) loss: 5.9681
(iter : 32800) loss: 5.5568
(iter : 32900) loss: 5.8729
(iter : 33000) loss: 6.3844
(iter : 33100) loss: 6.2450
(iter : 33200) loss: 5.6581
(iter : 33300) loss: 5.3461
(iter : 33400) loss: 6.1448
(iter : 33500) loss: 6.0203
(iter : 33600) loss: 5.2881
(iter : 33700) loss: 5.5329
(iter : 33800) loss: 6.1572
(iter : 33900) loss: 5.5910
(iter : 34000) loss: 5.3045
(iter : 34100) loss: 5.7411
(iter : 34200) loss: 6.0294
(iter : 34300) loss: 5.7551
(iter : 34400) loss: 5.7276
(iter : 34500) loss: 5.2570
(iter : 34600) loss: 6.4097
(iter : 34700) loss: 5.8104
(iter : 34800) loss: 5.6107
(iter : 34900) loss: 5.5763
(iter : 35000) loss: 5.9881
(iter : 35100) loss: 5.3598
(iter : 35200) loss: 5.8363
(iter : 35300) loss: 5.1685
(iter : 35400) loss: 5.4901
(iter : 35500) loss: 6.2269
(iter : 35600) loss: 5.7540
(iter : 35700) loss: 5.6384
(iter : 35800) loss: 5.7954
(iter : 35900) loss: 5.8762
(iter : 36000) loss: 5.4596
(iter : 36100) loss: 6.0190
(iter : 36200) loss: 5.4797
```

```
(iter : 36300) loss: 5.7611
(iter : 36400) loss: 5.6628
(iter : 36500) loss: 5.4364
(iter : 36600) loss: 6.1101
(iter : 36700) loss: 5.8566
(iter : 36800) loss: 5.6815
(iter : 36900) loss: 5.5675
(iter : 37000) loss: 5.4405
(iter : 37100) loss: 5.4371
(iter : 37200) loss: 5.7139
(iter : 37300) loss: 5.8147
(iter : 37400) loss: 5.9531
(iter : 37500) loss: 6.3149
(iter : 37600) loss: 5.5864
(iter : 37700) loss: 6.6413
(iter : 37800) loss: 6.4545
(iter : 37900) loss: 5.8501
(iter : 38000) loss: 5.6393
(iter : 38100) loss: 5.0896
(iter : 38200) loss: 5.5581
(iter : 38300) loss: 5.5386
(iter : 38400) loss: 4.9809
(iter : 38500) loss: 5.6937
(iter : 38600) loss: 5.7875
(iter : 38700) loss: 5.8045
(iter : 38800) loss: 5.4540
(iter : 38900) loss: 5.9911
(iter : 39000) loss: 4.9759
(iter : 39100) loss: 5.2068
(iter : 39200) loss: 5.7893
(iter : 39300) loss: 5.3823
(iter : 39400) loss: 5.8179
(iter : 39500) loss: 5.5376
(iter : 39600) loss: 5.9500
(iter : 39700) loss: 6.2621
(iter : 39800) loss: 5.2691
(iter : 39900) loss: 5.2749
(iter : 40000) loss: 5.8379
learning rate is decayed! current lr :  0.000625
model is saved!
(iter : 40100) loss: 5.6666
(iter : 40200) loss: 5.0203
(iter : 40300) loss: 5.4461
(iter : 40400) loss: 5.4882
(iter : 40500) loss: 5.2715
(iter : 40600) loss: 5.5353
(iter : 40700) loss: 5.6435
(iter : 40800) loss: 5.1290
(iter : 40900) loss: 5.0596
(iter : 41000) loss: 5.4713
(iter : 41100) loss: 5.5117
(iter : 41200) loss: 6.0590
(iter : 41300) loss: 5.1364
```

```
(iter : 41400) loss: 5.3703
(iter : 41500) loss: 4.9714
(iter : 41600) loss: 4.8057
(iter : 41700) loss: 4.8815
(iter : 41800) loss: 5.2175
(iter : 41900) loss: 5.3827
(iter : 42000) loss: 4.7106
(iter : 42100) loss: 5.8010
(iter : 42200) loss: 5.1960
(iter : 42300) loss: 5.0520
(iter : 42400) loss: 5.1033
(iter : 42500) loss: 4.5719
(iter : 42600) loss: 5.4548
(iter : 42700) loss: 5.4223
(iter : 42800) loss: 5.7508
(iter : 42900) loss: 5.3275
(iter : 43000) loss: 5.2812
(iter : 43100) loss: 5.2947
(iter : 43200) loss: 5.1553
(iter : 43300) loss: 5.3747
(iter : 43400) loss: 5.8999
(iter : 43500) loss: 5.3276
(iter : 43600) loss: 5.7653
(iter : 43700) loss: 4.8070
(iter : 43800) loss: 5.3410
(iter : 43900) loss: 5.5143
(iter : 44000) loss: 5.6690
(iter : 44100) loss: 4.6998
(iter : 44200) loss: 5.7023
(iter : 44300) loss: 5.8401
(iter : 44400) loss: 4.8846
(iter : 44500) loss: 5.1676
(iter : 44600) loss: 5.4089
(iter : 44700) loss: 6.0530
(iter : 44800) loss: 4.8464
(iter : 44900) loss: 5.5657
(iter : 45000) loss: 5.3389
(iter : 45100) loss: 5.1903
(iter : 45200) loss: 5.0831
(iter : 45300) loss: 5.2362
(iter : 45400) loss: 5.2488
(iter : 45500) loss: 5.3134
(iter : 45600) loss: 5.2800
(iter : 45700) loss: 5.1196
(iter : 45800) loss: 5.2331
(iter : 45900) loss: 5.3244
(iter : 46000) loss: 5.9557
(iter : 46100) loss: 5.0990
(iter : 46200) loss: 5.6907
(iter : 46300) loss: 5.3492
(iter : 46400) loss: 5.7573
(iter : 46500) loss: 5.1194
(iter : 46600) loss: 5.0296
```

```
(iter : 46700) loss: 4.8507
(iter : 46800) loss: 5.4791
(iter : 46900) loss: 5.0376
(iter : 47000) loss: 4.7316
(iter : 47100) loss: 5.4847
(iter : 47200) loss: 4.7419
(iter : 47300) loss: 5.1602
(iter : 47400) loss: 5.2695
(iter : 47500) loss: 5.6097
(iter : 47600) loss: 4.5336
(iter : 47700) loss: 5.4781
(iter : 47800) loss: 5.0843
(iter : 47900) loss: 5.5880
(iter : 48000) loss: 5.1140
(iter : 48100) loss: 4.8295
(iter : 48200) loss: 5.1302
(iter : 48300) loss: 5.5779
(iter : 48400) loss: 5.8680
(iter : 48500) loss: 4.8852
(iter : 48600) loss: 5.5230
(iter : 48700) loss: 4.8935
(iter : 48800) loss: 5.3663
(iter : 48900) loss: 4.9887
(iter : 49000) loss: 5.0624
(iter : 49100) loss: 5.4627
(iter : 49200) loss: 5.4449
(iter : 49300) loss: 4.5881
(iter : 49400) loss: 5.1473
(iter : 49500) loss: 4.9362
(iter : 49600) loss: 5.2656
(iter : 49700) loss: 5.3196
(iter : 49800) loss: 5.7607
(iter : 49900) loss: 5.1004
(iter : 50000) loss: 5.0850
learning rate is decayed! current lr :  0.0003125
model is saved!
(iter : 50100) loss: 5.0695
(iter : 50200) loss: 5.2510
(iter : 50300) loss: 5.0692
(iter : 50400) loss: 5.0668
(iter : 50500) loss: 4.9191
(iter : 50600) loss: 5.3715
(iter : 50700) loss: 5.5805
(iter : 50800) loss: 4.9803
(iter : 50900) loss: 4.9457
(iter : 51000) loss: 4.5678
(iter : 51100) loss: 5.1060
(iter : 51200) loss: 5.0562
(iter : 51300) loss: 5.0572
(iter : 51400) loss: 5.3131
(iter : 51500) loss: 4.9337
(iter : 51600) loss: 5.6328
(iter : 51700) loss: 5.1947
```

```
(iter : 51800) loss: 5.0602
(iter : 51900) loss: 5.5297
(iter : 52000) loss: 4.9329
(iter : 52100) loss: 4.7286
(iter : 52200) loss: 4.8670
(iter : 52300) loss: 4.8244
(iter : 52400) loss: 5.6606
(iter : 52500) loss: 4.9943
(iter : 52600) loss: 5.1817
(iter : 52700) loss: 4.9566
(iter : 52800) loss: 4.5835
(iter : 52900) loss: 4.8447
(iter : 53000) loss: 4.4857
(iter : 53100) loss: 4.7622
(iter : 53200) loss: 4.8365
(iter : 53300) loss: 5.0940
(iter : 53400) loss: 5.5394
(iter : 53500) loss: 5.0838
(iter : 53600) loss: 5.1202
(iter : 53700) loss: 4.6518
(iter : 53800) loss: 4.9755
(iter : 53900) loss: 4.8903
(iter : 54000) loss: 4.5273
(iter : 54100) loss: 4.8837
(iter : 54200) loss: 5.8058
(iter : 54300) loss: 5.0808
(iter : 54400) loss: 5.2725
(iter : 54500) loss: 4.8350
(iter : 54600) loss: 5.5083
(iter : 54700) loss: 4.7948
(iter : 54800) loss: 5.1863
(iter : 54900) loss: 5.6006
(iter : 55000) loss: 4.6568
(iter : 55100) loss: 5.0928
(iter : 55200) loss: 4.9535
(iter : 55300) loss: 5.2271
(iter : 55400) loss: 6.0313
(iter : 55500) loss: 5.6088
(iter : 55600) loss: 5.1145
(iter : 55700) loss: 5.1067
(iter : 55800) loss: 5.0314
(iter : 55900) loss: 4.4354
(iter : 56000) loss: 5.3336
(iter : 56100) loss: 4.6303
(iter : 56200) loss: 5.4911
(iter : 56300) loss: 4.8011
(iter : 56400) loss: 4.7278
(iter : 56500) loss: 4.3644
(iter : 56600) loss: 4.8671
(iter : 56700) loss: 4.1991
(iter : 56800) loss: 5.0187
(iter : 56900) loss: 4.8290
(iter : 57000) loss: 5.2489
```

```
(iter : 57100) loss: 4.8422
(iter : 57200) loss: 4.7432
(iter : 57300) loss: 5.2697
(iter : 57400) loss: 4.3247
(iter : 57500) loss: 5.0200
(iter : 57600) loss: 4.9058
(iter : 57700) loss: 4.7815
(iter : 57800) loss: 5.0891
(iter : 57900) loss: 5.6554
(iter : 58000) loss: 4.7796
(iter : 58100) loss: 4.6124
(iter : 58200) loss: 4.3885
(iter : 58300) loss: 5.4216
(iter : 58400) loss: 5.0885
(iter : 58500) loss: 5.1005
(iter : 58600) loss: 4.5721
(iter : 58700) loss: 4.9611
(iter : 58800) loss: 5.1177
(iter : 58900) loss: 5.3826
(iter : 59000) loss: 5.0858
(iter : 59100) loss: 5.3879
(iter : 59200) loss: 5.5520
(iter : 59300) loss: 4.6062
(iter : 59400) loss: 4.9064
(iter : 59500) loss: 5.0871
(iter : 59600) loss: 4.8817
(iter : 59700) loss: 5.1417
(iter : 59800) loss: 5.0438
(iter : 59900) loss: 5.2398
(iter : 60000) loss: 4.5126
learning rate is decayed! current lr :  0.00015625
model is saved!
```

```
config.train = False
test("./model")
```

```
embedded size:  (160, 64)
model path : ./model
ckpt file is loaded ! ./model\Check_Point\model.ckpt-5
INFO:tensorflow:Restoring parameters from ./model\Check_Point\model.ckpt-5
test file path :  ./test_tisv
inference time for 160 utterences : 0.59s
[[[ 0.84  0.25 -0.21  0.33  0.12  0.25  0.49  0.11]
  [ 0.85  0.53 -0.42  0.47  0.21  0.44  0.34 -0.02]
  [ 0.88  0.41 -0.15  0.34  0.28  0.47  0.36  0.08]
  [ 0.81  0.55 -0.15  0.43  0.22  0.45  0.22  0.07]
  [ 0.74  0.08 -0.07  0.53  0.03  0.18  0.36  0.01]
  [ 0.8   0.39  0.06 -0.06  0.32  0.33  0.51  0.21]
  [ 0.79  0.35 -0.28  0.21  0.24  0.36  0.16  0.07]
  [ 0.64  0.42  0.12 -0.12  0.33  0.47  0.21  0.21]
```

```
  [ 0.36  0.18  0.19 -0.05  0.53  0.24 -0.08  0.07]
  [ 0.68  0.59 -0.34  0.38  0.29  0.56 -0.01 -0.12]]

 [[ 0.34  0.79 -0.58  0.14 -0.12  0.69 -0.14 -0.44]
  [ 0.28  0.74  0.14 -0.02  0.39  0.63 -0.19  0.  ]
  [ 0.3   0.77 -0.34  0.29  0.    0.72 -0.22 -0.6 ]
  [ 0.12  0.86 -0.16 -0.04  0.02  0.74 -0.19 -0.18]
  [ 0.54  0.79 -0.49  0.4   0.03  0.72  0.04 -0.33]
  [ 0.    0.7   0.08 -0.22 -0.    0.62 -0.19  0.09]
  [ 0.6   0.91 -0.41  0.29  0.06  0.79 -0.04 -0.28]
  [ 0.45  0.81 -0.22 -0.02  0.1   0.67  0.05  0.17]
  [ 0.54  0.91 -0.18  0.11  0.16  0.84 -0.14 -0.11]
  [ 0.43  0.75 -0.36  0.08  0.    0.68 -0.13  0.11]]

 [[-0.05 -0.55  0.84 -0.05  0.04 -0.45  0.22  0.47]
  [ 0.01 -0.46  0.95 -0.14  0.11 -0.33  0.34  0.42]
  [-0.07 -0.46  0.98 -0.02 -0.03 -0.41  0.34  0.39]
  [-0.05 -0.38  0.99 -0.12  0.   -0.3   0.3   0.47]
  [-0.12 -0.33  0.97 -0.11  0.04 -0.27  0.18  0.28]
  [-0.08 -0.35  0.96 -0.05  0.02 -0.25  0.15  0.32]
  [-0.11 -0.49  0.96 -0.09 -0.   -0.39  0.22  0.41]
  [ 0.02 -0.46  0.94 -0.11  0.15 -0.36  0.36  0.41]
  [ 0.18 -0.29  0.8  -0.03  0.02 -0.29  0.51  0.44]
  [-0.11 -0.43  0.92 -0.24  0.03 -0.27  0.2   0.51]]

 [[ 0.33  0.31 -0.32  0.93 -0.4   0.19  0.26 -0.26]
  [ 0.55  0.2   0.02  0.82 -0.07  0.11  0.31 -0.14]
  [ 0.46  0.28 -0.26  0.94 -0.31  0.16  0.4  -0.17]
  [ 0.34  0.24 -0.1   0.91 -0.23  0.07  0.28 -0.14]
  [ 0.16  0.2  -0.06  0.86 -0.43  0.09  0.13 -0.28]
  [ 0.48  0.07  0.06  0.93 -0.26  0.01  0.39  0.01]
  [ 0.23 -0.04  0.07  0.9  -0.28 -0.08  0.21 -0.16]
  [ 0.14  0.1   0.2   0.77 -0.13  0.05  0.04 -0.16]
  [ 0.53  0.39 -0.43  0.87 -0.34  0.28  0.31 -0.2 ]
  [ 0.5   0.37 -0.36  0.86 -0.43  0.28  0.31 -0.16]]

 [[ 0.22  0.18  0.05 -0.33  0.92  0.14 -0.07 -0.18]
  [ 0.26  0.2   0.05 -0.42  0.92  0.17 -0.06 -0.05]
  [ 0.29  0.03  0.15 -0.37  0.81  0.07  0.05 -0.12]
  [ 0.19  0.15  0.15 -0.39  0.94  0.15 -0.07 -0.07]
  [ 0.12 -0.05 -0.24 -0.29  0.89 -0.12 -0.06 -0.21]
  [-0.03  0.08  0.07 -0.34  0.85  0.07 -0.21 -0.4 ]
  [-0.    0.02 -0.04 -0.35  0.88 -0.   -0.17 -0.39]
  [ 0.2   0.18  0.16 -0.31  0.88  0.13 -0.08 -0.11]
  [ 0.09  0.05 -0.14 -0.32  0.94 -0.03 -0.15 -0.27]
  [ 0.15  0.06 -0.11 -0.31  0.95 -0.03 -0.09 -0.2 ]]

 [[ 0.48  0.65 -0.59  0.33  0.03  0.61 -0.03 -0.32]
  [ 0.1   0.8  -0.2  -0.18  0.03  0.88 -0.45 -0.29]
  [ 0.41  0.76 -0.65  0.23 -0.07  0.72 -0.14 -0.44]
  [ 0.52  0.73 -0.59  0.22 -0.    0.67  0.02 -0.33]
  [-0.07  0.54  0.23 -0.11  0.1   0.51 -0.25 -0.32]
  [ 0.2   0.69  0.19 -0.17  0.31  0.68 -0.26 -0.13]
```

```
  [ 0.44  0.37 -0.28  0.64 -0.26  0.33 -0.01 -0.22]
  [ 0.1   0.68 -0.   -0.25  0.12  0.75 -0.35 -0.13]
  [ 0.3   0.39 -0.28  0.71 -0.34  0.44 -0.05 -0.11]
  [ 0.02  0.3   0.05 -0.25  0.2   0.5  -0.4   0.16]]

 [[ 0.46 -0.36  0.21  0.35  0.01 -0.51  0.89  0.26]
  [ 0.4  -0.12  0.51 -0.05  0.2  -0.26  0.83  0.42]
  [ 0.2  -0.4   0.31  0.24  0.07 -0.58  0.81  0.26]
  [ 0.47 -0.03  0.36  0.22  0.03 -0.21  0.91  0.35]
  [ 0.43 -0.39  0.32  0.44 -0.11 -0.46  0.9   0.23]
  [ 0.46 -0.13  0.33  0.33  0.09 -0.28  0.92  0.23]
  [ 0.51 -0.19  0.11  0.37  0.05 -0.28  0.92  0.2 ]
  [ 0.38 -0.15  0.36  0.2   0.08 -0.3   0.86  0.33]
  [ 0.25 -0.12  0.08  0.35 -0.04 -0.33  0.77  0.1 ]
  [ 0.5  -0.08  0.23  0.3   0.03 -0.22  0.92  0.22]]

 [[ 0.07 -0.1   0.41  0.08 -0.39 -0.18  0.18  0.86]
  [ 0.08 -0.12  0.55 -0.31  0.08 -0.14  0.27  0.89]
  [ 0.24 -0.09  0.18 -0.02 -0.13 -0.11  0.16  0.86]
  [ 0.23 -0.16  0.27 -0.04 -0.17 -0.16  0.24  0.9 ]
  [ 0.2  -0.08  0.47 -0.37  0.15 -0.14  0.42  0.76]
  [ 0.12 -0.38  0.56 -0.24  0.2  -0.42  0.43  0.78]
  [-0.09 -0.32  0.2   0.15 -0.31 -0.32 -0.04  0.73]
  [ 0.14 -0.15  0.58 -0.35  0.23 -0.15  0.29  0.79]
  [ 0.01 -0.18  0.45  0.03 -0.43 -0.18  0.26  0.74]
  [ 0.27 -0.12  0.54 -0.17  0.11 -0.21  0.53  0.87]]]

EER : 0.06 (thres:0.52, FAR:0.06, FRR:0.06)
```