语音信号处理实验报告

课程名称：语音信号处理　　实验日期：2023/11/2

班级：人工智能 3 班　　姓名：啊啊啊　　学号：啊啊啊啊 6

实验四 声纹识别系统实现及验证

实践一：基于 x-vector 的声纹识别系统的实现与验证

一. 实践要求

实现基于 x-vector 的声纹识别系统的训练。了解声纹识别系统训练的具体流程以及深度神经网络等重要组件。以及实现基于 x-vector 的声纹识别系统的验证。了解声纹识别系统验证的具体流程以及 EER 等评分方式的计算。

二. 实践内容

1. 理解并分析声纹识别系统训练的具体流程；
2. 了解 Softmax 损失函数并补全代码；
3. 基于 x-vector 的声纹识别系统的训练模块分析；
4. 理解并分析声纹识别系统验证的具体流程；
5. 导入训练的模型并计算余弦相似度；
6. 绘制 ROC 曲线并分析等错误率（EER）的计算流程。

三. 实践结果与分析

1. 理解并分析声纹识别系统训练的具体流程

(1) 数据配置部分

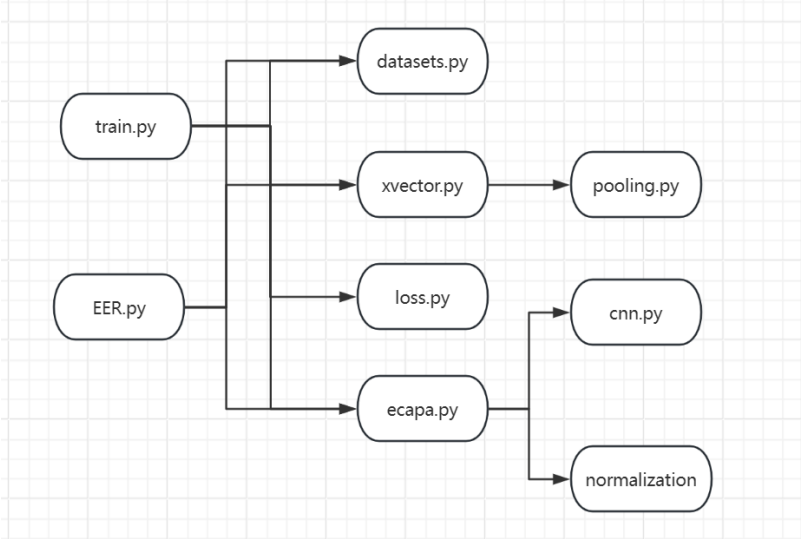
```
data:
  train_manifest: files\split_train_set.txt
  # ==========> code <============
  root: Split
  # ==========> code <============
  epoch: 80
  feat_type: mfcc
```

训练部分，root 是相对路径，本地下载的数据集，因为我自己练了一个所以

改了 epoch。

```
test:
  # model_path: saved_model\000\checkpoint_100.pth
  model_path: saved_model\xvector_gap_softmax_checkpoint.pth
  test_manifest: files\test_set.txt
```

测试的时候会用自己练得和老师给的两个模型跑一边看数。



这是项目文件继承图。

训练代码补全如下，采用 softmax 计算损失函数，学习器 adam

```
# ====================> code <========================
# 定义模型
model = Xvector().to(device)

# 定义损失函数
criterion = Softmax(512, args.speaker_size).to(device)

# 定义学习器
optimizer = optim.Adam(params=model.parameters(), lr=1e-5)

# ====================> code <========================
```

2. 了解 Softmax 损失函数并补全代码

针对 softmax 损失函数的前向计算，第一步计算全连接层，然后求交叉熵损失，并返回准确度和 loss

```python
def forward(self, embeddings, labels):
    # ==========> code <===========
    logits = self.fc(embeddings)
    loss = F.cross_entropy(logits, labels)
    # ==========> code <===========
    acc = self.accuracy(logits, labels)
    return loss, acc
```

3. 基于 x-vector 的声纹识别系统的训练模块分析

声纹识别 Xvector 首先采用五层卷积，最后进行全平均池化，然后全连接层，此处多了一个拉平操作，因为不拉平会报错 batch 数与 weight 矩阵不等。

```python
    # ==========> code <===========
    self.blocks.extend([
        # global average pooling
        GlobalAveragePooling(),
        # Final linear transformation
        nn.Flatten(),
        nn.Linear(1500, lin_neurons)
    ])
    # ==========> code <===========
    # print(self)
```

单模块测试如下，运行良好

```python
if __name__ == '__main__':
    model = Xvector()
    input_feats = torch.rand([64, 23, 100])
    outputs: torch.Tensor = model(input_feats)
    print(outputs.shape)
```

```
(spkcls) D:\1-School\语音信号处理\04
Verification_Code/xvector.py
torch.Size([64, 512])
```

训练部分补全 train 代码，然后进行计算

```python
  # ==================> code <======================
  outputs = model(feats)
  loss, acc = criterion(outputs, labels)
  loss.backward()
  optimizer.step()
  optimizer.zero_grad()
  # ==================> code <======================
```

练到这个程度就行了，再高就过拟合了

```
Train Stage: 79/41 ===>  Loss 2.220      Acc 0.883
Train Stage: 79/42 ===>  Loss 2.220      Acc 0.883
Train Stage: 79/43 ===>  Loss 2.220      Acc 0.883
Train Stage: 79/44 ===>  Loss 2.220      Acc 0.882
Train Stage: 79/45 ===>  Loss 2.224      Acc 0.880
Train Stage: 79/46 ===>  Loss 2.224      Acc 0.880
```

4. 理解并分析声纹识别系统验证的具体流程；

训练部分再跑一次，model.train 改成 model.eval 基本就是验证

5. 导入训练的模型并计算余弦相似度；

```
# Load model from checkpoint
# ==================> code <======================
# =============== 定义并导入模型 ==================
model = Xvector().to(device)
model_dict = torch.load(log_dir)
model.load_state_dict(model_dict, strict=False)
# ==================> code <======================
print('eval')
model.eval()
```

调用 torch 接口载入模型

```
feats = feats.to(device)
# ==================> code <======================
# ============ 为测试数据生成深度嵌入 ==============
embeddings = model(feats)
# ==================> code <======================

for i in range(embeddings.shape[0]):
```

计算输出嵌入值

```
with torch.no_grad():
    enroll_embedding = dict_embedding[enroll_filename].unsqueeze(0)
    test_embedding = dict_embedding[test_filename].unsqueeze(0)

    # ==================> code <======================
    # ================ 计算相似度 ====================
    score = F.cosine_similarity(enroll_embedding, test_embedding)
    # ==================> code <======================
    score = score.data.cpu().numpy()[0]
    del enroll_embedding
    del test_embedding
# print(score, label)
score_list.append(score)
```

计算嵌入值和真实值的余弦相似度

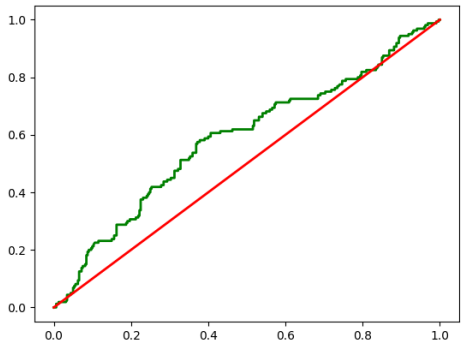
6. 绘制 ROC 曲线并分析等错误率（EER）的计算流程。

使用强大的 sklearn.metrics 库进行 roc 曲线计算，并分析错误率 EER

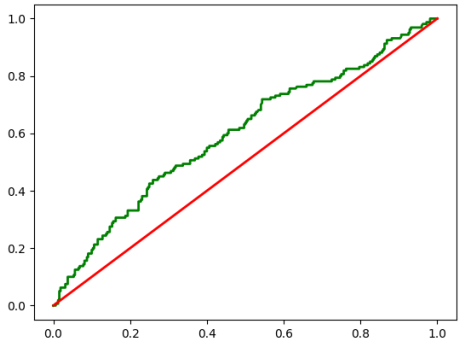
```python
from sklearn.metrics import roc_curve, auc
def roc(score_list, label_list):
    # ==================> code <======================
    # ================ 绘制roc曲线 ===================
    # 计算ROC各点
    fpr, tpr, thresholds = roc_curve(label_list, score_list)
    # 绘制ROC曲线
    plt.plot(fpr, tpr, linewidth=2, color='g', label='ROC Curve (area = %0.2f)' % auc(fpr, tpr))
    plt.plot([0, 1], [0, 1], linewidth=2, color='r')
    plt.show()
    # pass
    # ==================> code <======================
```

错误率分析还得是先计算 roc 曲线，然后根据阈值求相差的绝对值

```python
def get_eer(score_list, label_list):
    fpr, tpr, threshold = roc_curve(label_list, score_list, pos_label=
    fnr = 1 - tpr
    eer_threshold = threshold[np.nanargmin(np.absolute((fnr - fpr)))]
    eer = fpr[np.nanargmin(np.absolute((fnr - fpr)))]
    intersection = abs(1 - tpr - fpr)
    print("Epoch=%d  EER= %.2f  Thres= %0.5f" % (
    args.cp_num, 100 * fpr[np.argmin(intersection)], eer_threshold))
    return eer, eer_threshold
```



最后对比我的模型和老师的模型



左图是 xvector_gap_softmax_checkpoint，右图是我自己练的 checkpoint_020，学习率 1e-4，20 轮迭代，感觉发挥是相近的，而且 roc 曲线的计算并不稳定，每次都不一样，这是两次最好的结果。

实践心得

语音信号真好玩