因诺微科技(天津)有限公司 OLR-AP19 比赛方案介绍

1. 任务描述
2. 数据增广
3. 模型描述
4. 特征描述
5. 任务域自适应方法
6. Crop-N 测试方法
7. 任务实现方案
8. 模型融合

方案概况

1 对数据进行多种方式随机增强

2 对任务 1 进行 12 个模型训练

12 个模型: 3 个模型结构 x 4 个特征

3 对任务 2 任务 3 使用任务 1pre-trained 模型 进行任务域自适应得到对应的

4 对每个任务进行 Crop-N 测试方法进行测试

5 12 个模型按照训练准确率进行加权平均.

1 任务描述

task_1　1s 识别
特点: 只以 1s 语音段为输入进行语种识别, 输入语音短, 需要抽取区分能力特征
方案: 使用 softmax，而不使用 embedding 方法.

task_2　跨信道识别
特点: 训练数据与测试集合存在信道差异, 使用原始模型测试效果差
方案: 进行域自适应方法, 补偿模型对测试集合的失配问题.

task_3　零资源识别
特点: 测试集合与训练集合完全没有交集, 训练得到的模型无法直接用
方案: 扩展域自适应方法, 将域自适应方法扩展为任务域自适应方法

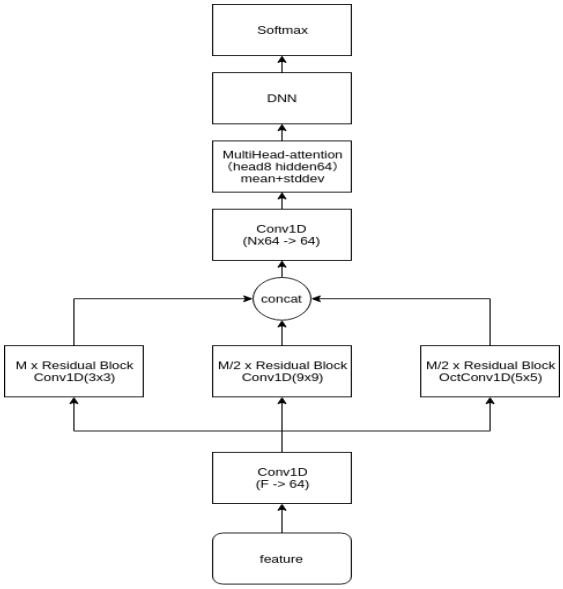
（组委会注：2、3 中使用测试集对系统进行再训练，与比赛要求不符，故此处方案仅做参考。）

2 数据增广

各种数据增广处理方法:
'no', 无处理
'speed', 速度扰动
'volume', 音量扰动
'add-w-noise', 增加白噪声
'pitch', pitch 扰动
'stft', 频域随机变换
'vad', 静音去除

3 模型描述

3.1 TResNet



1 将不同种类特征维度利用 Conv1D(3x3)+maxpool(2x2)映射为 64 维 base feature

2 多路 ResidualBlock Conv1D

low path feature

middel path feature

high path feature

3 不同模型(TResNet-A TResNet-B TResNet-C)拼接不同 path feature 输出 N path feature 维度为 N*64

4 将 N path feature 利用 Conv1D(3x3)+maxpool(2x2) 映射为 64 维 out feature

5 经过 multihead-attention(head=8, hidden=64),

6 使用 mean+stddev 获得 128 维的特征向量

7 DNN + softmax(10) 进行识别

激活函数选择为 PRelu

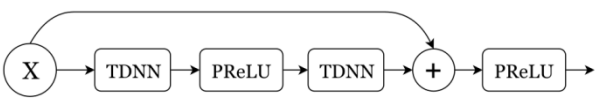
ResidualBlock

其中 TDNN 我们选择三种方案

1 Conv1D, kernel=(3x3), stride=(1x1)

2 Conv1D, kernel=(9x9), stride=(1x1)

3 OctConv1D, kernel=(5x5), stride=(1x1)



TResNet-A（未使用）

只使用 Conv1D(3x3) ResidualBlock, M=20

TResNet-B
使用 Conv1D(3x3) ResidualBlock, M=20
使用 Conv1D(9x9) ResidualBlock, M=10
拼接两个 ResidualBlock sequantial 的输出

TResNet-C
使用 Conv1D(3x3) ResidualBlock, M=20
使用 Conv1D(9x9) ResidualBlock, M=10
使用 OctConv1D(5x5) ResidualBlock, M=10
拼接三个 ResidualBlock sequantial 的输出

3.2 EfficientNet 调参

efficientnet_params 调参

'efficientnet-lid': (1.2, 1.4, 300, 0.2)

```
blocks_args = [
            'r1_k3_s11_e1_i64_o64_se0.25',
            'r2_k3_s11_e6_i64_o64_se0.25',
            'r2_k5_s22_e6_i64_o128_se0.25',
            'r2_k3_s11_e6_i128_o128_se0.25',
            'r2_k5_s22_e6_i128_o64_se0.25',
            'r3_k3_s11_e6_i64_o64_se0.25',
            'r2_k3_s11_e6_i64_o64_se0.25',
      ]
```

在 EfficientNet 前端对不同输入特征进行归一化，使用 Conv2D 将特征都变换到 64 维

最终仍然使用 softmax 进行分类

4 特征描述

FFT

10ms 帧移, 20ms 帧窗，采样率 8k

增加前后 delta

特征维度 243

FDLP

频域线性预测系数

使用 TAM 方法

特征维度 13

SincF

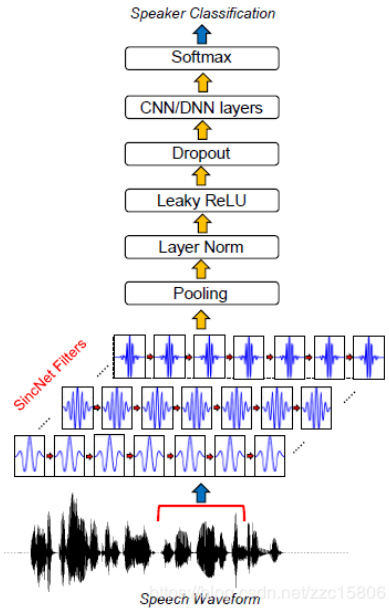
以 signal 为输入

用 SincNet 的前端部分作特征提取得到 SincFeature

特征维度 96

取其中 SincNet Filters 部分

作为特征 SincF 的提取器，与模型联合训练.



MPIF

Minimum Phase Instantaneous Frequency 最小相位瞬时频率

特征维度 37

5 任务域自适应方法

1 认为模型在环境适应情况下识别完美.
2 导致不完美的情况是由于域失配造成的.
3 将测试置信度较高的测试样本，为其设置 one-hot 标签，进行反向传播更新模型参数

第一阶段

用任务 1pre-trained 模型，对注册数据进行训练达到稳定，得到任务 2pre-trained 模型.

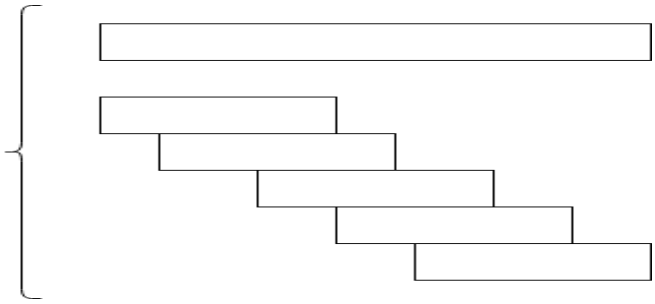
第二阶段

用任务 2pre-trained 模型，对任务测试数据进行打分，将每个语种得分较高的 percent 数据，纳入训练，达到稳定

重复第二阶段三次，percent 分别为 10% 40% 90% 实现任务域自适应方法，得到任务 2 模型

6 Crop-N 测试方法

1 将一个语音段，以有重叠方式进行分割为 N 个固定长度的语音段
2 每个语音段都进行测试得到概率向量
3 对多个概率向量取平均 得到 result



7 任务实施方案

任务 1 方案

因为语种数量本身较少，不存在分类稀疏问题，使用 softmax 实现 End2End.

因为任务语音只有 1s，使用的训练数据也限定为只使用 1s.

多模型融合，选择使用 12 个模型进行融合

测试数据只有 1s，直接测试.

任务 2 方案

模型直接来自 Task_1 pre-trained Model.

使用验证数据进行第一阶段再训练

使用任务域自适应方法进行第二阶段适应性再训练.

Crop-5 进行测试

任务 3 方案

模型直接来自 Task_1 pre-trained Model.

使用提供的注册样本，进行第一阶段再训练.

使用任务域自适应方法进行第二阶段适应性再训练.

Crop-10 进行测试

8 模型融合方法

每个模型的训练准确率为权重，进行加权平均

	TResNet-B	TResNet-C	EfficientNet-m
FFT			
FDLP			
SincF			
MPIF			

References

[1] “KingLine Data Center, AP16-OL7 Multilingual Database, Speechocean Ltd. (www.speechocean.com), 2016.”

[2] “Zhiyuan Tang, Dong Wang, Liming Song: AP19-OLR Challenge: Three Tasks and Their Baselines, APSIPA ASC 2019.”

[3] Sergey Novoselov, Andrey Shulipa : On deep speaker embeddings for text-independent speaker recognition

[4] Sarith Fernando, Vidhyasaharan Sethu, Eliathamby Ambikairajah: Sub-band Envelope Features using Frequency Domain Linear Prediction for Short Duration Language Identification

[5] Sarith Fernando: Deep Learning Approaches to Feature Extraction, Modelling and Compensation for Short Duration Language Identification

[6] Mingxing Tan, Quoc V. Le: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

[7] Ashish Vaswani . et: Attention Is All You Need

[8] Yunpeng Chen , Haoqi Fang , Bing Xu: Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution

[9] Mirco Ravanelli，Yoshua Bengio: Interpretable Convolutional Filters with SincNet