

# Espnet实践

洪青阳

# Espnet介绍

2018年，Espnet团队开源了Espnet（end-to-end speech processing toolkit），可实现端到端ASR和TTS系统。

## [Espnet特色]

- (1) 融合了Kaldi的数据处理和特征提取；
- (2) 借助Pytorch跟Chainer，使用Python实现了端到端（E2E）模型；

## [Espnet模型]

- CTC
- Attention
- RNN-T
- Transformer

# Espnet的安装

## 一、准备工作

1. 用git的方法把Espnet和Kaldi的项目保存到本地。
2. 安装CUDA和对应版本的CUDADNN。
3. 安装Pytorch, 要和CUDA版本对应。

## 二、安装过程

1. 设置CUDA的环境变量路径（`~/.bashrc`），安装指导里有代码，只要修改第一行代码为CUDA实际安装路径就好了，设置完之后要执行`source ~/.bashrc`让它生效。
2. 安装Kaldi（参照第13章）。
3. 把Kaldi工具导入到Espnet项目中去。  
进入Espnet中的tools文件夹，执行命令：  
`make KALDI=/path_to_kaldi`  
同样可以执行`make check_install`检查安装。
4. 安装完成后，把项目放到Espnet里的egs目录下就可以用了。

# EspNet训练和测试步骤

用于训练的数据由两部分组成:

- 音频文件
- 标注文本 (transcriptions)  
格式: [ id words ]

02

语言模型: RNN-LM  
声学模型: Transformer的解码器和CTC模型均会根据输入序列 $X^{\text{fbank}}$ , 计算目标序列 $Y$ 的后验概率:  $p_{\text{tran}}(Y|X)$  和  $p_{\text{ctc}}(Y|X)$   
损失函数为负对数似然的加权和:

$$L = -\alpha \log p_{\text{tran}}(Y|X) - (1 - \alpha) \log p_{\text{ctc}}(Y|X)$$

04



数据预处理

01



特征提取

工具: Kaldi  
特征选择: 提取80维的FBank, 加上3维的pitch, 然后进行倒谱均值归一化, 目的是让神经网络更容易对语音特征进行学习。



模型训练

03



解码

根据语音特征序列 $X^{\text{fbank}}$ 和之前预测出的tokens, 使用剪枝搜索 (Beam Search) 算法。

联合Transformer模型、CTC模型和RNN语言模型进行打分:

$$\hat{Y} = \underset{Y \in \mathcal{Y}^*}{\operatorname{argmax}} \{ \lambda \log p_{\text{tran}}(Y|X) + (1 - \lambda) \log p_{\text{ctc}}(Y|X) + \gamma \log p_{\text{lm}}(Y) \}$$

# 训练数据准备

## 1. 数据集的整理

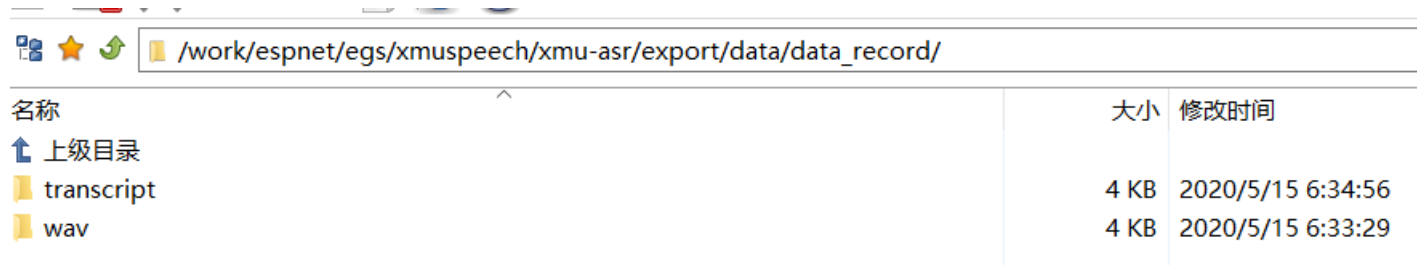
数据下载: <https://pan.xmu.edu.cn/s/MsUNwn3ASHo> 密码: 0o8o

数据集存放在export/data/data\_record/wav中, 共有62个说话人, 每个人大约有350~400句语音, 个别的少于350或多于400句。每一个说话人用不同ID (如S005) 标识, 每个文件夹中存放着对应于标注文本的音频文件。

最终整理后的数据集有2个文件夹:

a) transcript

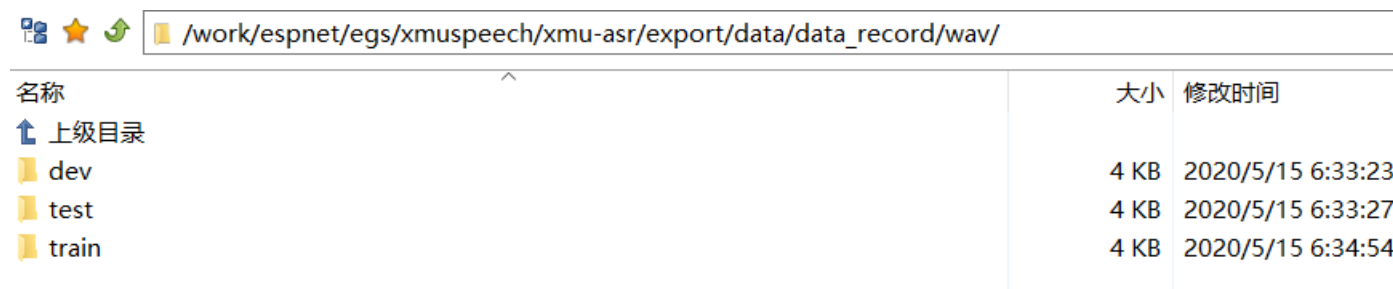
b) wav



/work/espnet/egs/xmuspeech/xmu-asr/export/data/data_record/		
名称	大小	修改时间
上级目录		
transcript	4 KB	2020/5/15 6:34:56
wav	4 KB	2020/5/15 6:33:29

wav数据集的划分:

train: 53人 dev: 6人 test: 3人

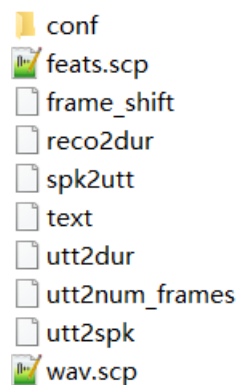


/work/espnet/egs/xmuspeech/xmu-asr/export/data/data_record/wav/		
名称	大小	修改时间
上级目录		
dev	4 KB	2020/5/15 6:33:23
test	4 KB	2020/5/15 6:33:27
train	4 KB	2020/5/15 6:34:54

# 训练数据准备

## 2. 映射文件准备

需要准备wav.scp、spk2utt、utt2spk、text这几个映射文件，默认已处理好。如有新的数据，可使用aishell的recipe中提供的aishell\_data\_prep.sh脚本，新建一个项目，修改data\_prep脚本并执行，便得到了相关的映射文件。



- conf
- feats.scp
- frame\_shift
- reco2dur
- spk2utt
- text
- utt2dur
- utt2num\_frames
- utt2spk
- wav.scp

4 KB	2020/5/27 23:22:10
1.76 MB	2020/5/27 23:22:09
5	2020/5/27 23:22:10
461 KB	2020/5/27 23:22:23
353 KB	2020/5/27 23:22:11
979 KB	2020/5/27 22:59:55
461 KB	2020/5/27 23:22:10
445 KB	2020/5/27 23:22:09
446 KB	2020/5/27 23:22:11
1.51 MB	2020/5/27 22:59:55

# 训练数据准备

## 3. 特征提取

调用了Espnet中默认使用的make\_fbank\_pitch.sh 进行特征提取，该脚本提取了80维的FBank特征，加上3维的pitch特征，总共83维。

特征提取完毕后，使用Kaldi的命令compute-cmvn-stats来进行CMVN（倒谱均值归一化）的计算，使神经网络更容易对语音特征进行学习。

特征提取结束后，得到了feats.scp文件，该文件保存了语音特征及其对应位置，如下图所示：

```
RECORD2020S004W001 /home/qwq/espnet/egs/record/asr1/fbank/raw_fbank_pitch_train.1.ark:19  
RECORD2020S004W002 /home/qwq/espnet/egs/record/asr1/fbank/raw_fbank_pitch_train.1.ark:39401  
RECORD2020S004W003 /home/qwq/espnet/egs/record/asr1/fbank/raw_fbank_pitch_train.1.ark:81107
```

# 训练数据准备

## 4. 词典生成

这一步是将字符转换为特定的数字，从而将每条语音对应的转录文本映射为不同的数字，以便送入神经网络中进行相应的训练。

在Espnet中，使用脚本text2token.py来通过映射文件中的text文件来生成词典，得到每个字符的唯一标识数字，词典文件如下图所示：

```
<unk> 1  
一 2  
丁 3  
七 4  
万 5  
丈 6  
三 7  
上 8  
下 9  
不 10  
与 11  
丑 12  
专 13  
且 14  
世 15
```



# 训练数据准备

## 5. 数据打包

在Espnet中训练神经网络，不是直接使用feat.scp或者text这些映射文件，而是使用脚本data2json.sh将这些文件都打包到一个json文件中，整体结构分为两个部分：input 和 output, input 对应于该条语音的特征以及特征的shape（表示维度），output 对应于该条语音的文本及其数字表示。

格式如下图：

```
"utts": {  
    "sp0.9-RECORD2020S004W001": {  
        "input": [  
            {  
                "feat": "/work/espnet/egs/xmuspeech/xmu-asr/dump/train_sp/deltafalse/feats.1.ark:25",  
                "name": "input1",  
                "shape": [  
                    518,  
                    83  
                ]  
            }  
        ],  
        "output": [  
            {  
                "name": "target1",  
                "shape": [  
                    13,  
                    1889  
                ],  
                "text": "那时的东西都很美时间都很慢",  
                "token": "那 时 的 东 西 都 很 美 时 间 都 很 慢",  
                "tokenid": "1706 860 1201 17 1532 1715 649 1367 860 1760 1715 649 708"  
            }  
        ],  
        "utt2spk": "sp0.9-s004"  
    },  
    "sp0.9-RECORD2020S004W002": {
```

# 模型的训练和解码

通过以上五个步骤，就完成了训练数据的准备工作。

接下来可运行run.sh，进行模型的训练和解码。

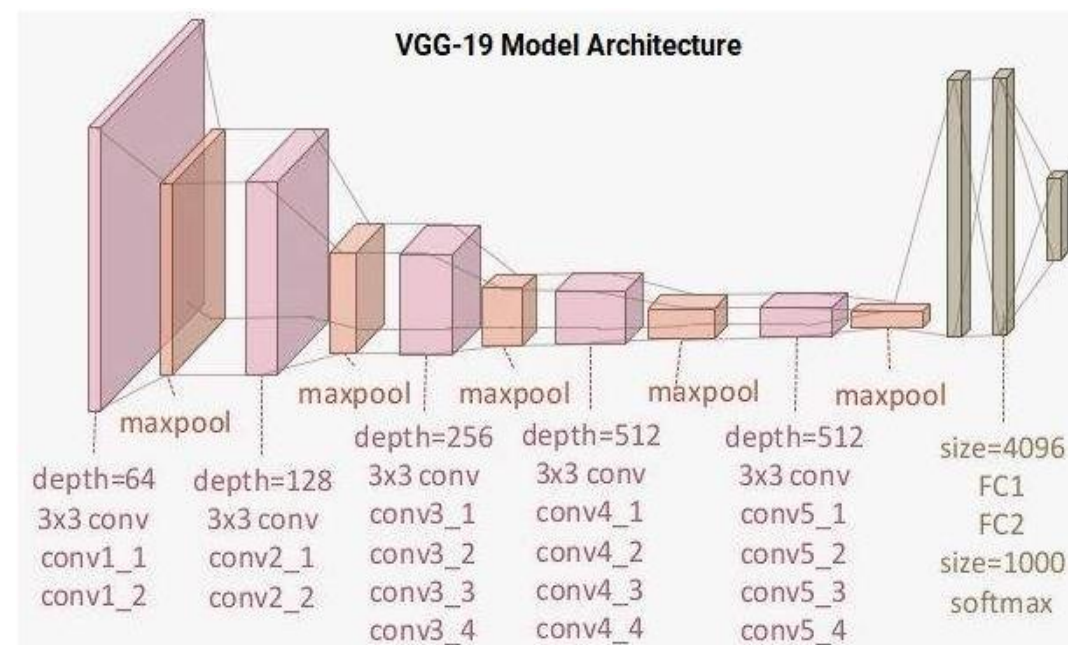
声学模型

- 1.CTC模型： 声学编码器+CTC损失函数
- 2.Attention模型： Espnet提供不同的Attention架构
- 3.Transformer： 并行计算的特征提取器
- 4.RNN-T： 适合流识别的语音识别模型
- 5.Transformer-T： 新结构，借助Transformer的优势来改善RNN-T模型的性能
- 6.CTC+Attention结构： 利用CTC来辅助Attention模型学习文本到语音的对齐

# EspNet中的声学模型

CTC

```
# network architecture
# encoder related
etype: vggblstm # encoder architecture type
elayers: 3
eunits: 1024
eprojs: 1024
subsample: "1_2_2_1_1" # skip every n frame from input to nth
layers
# decoder related
dlayers: 2
dunits: 1024
```



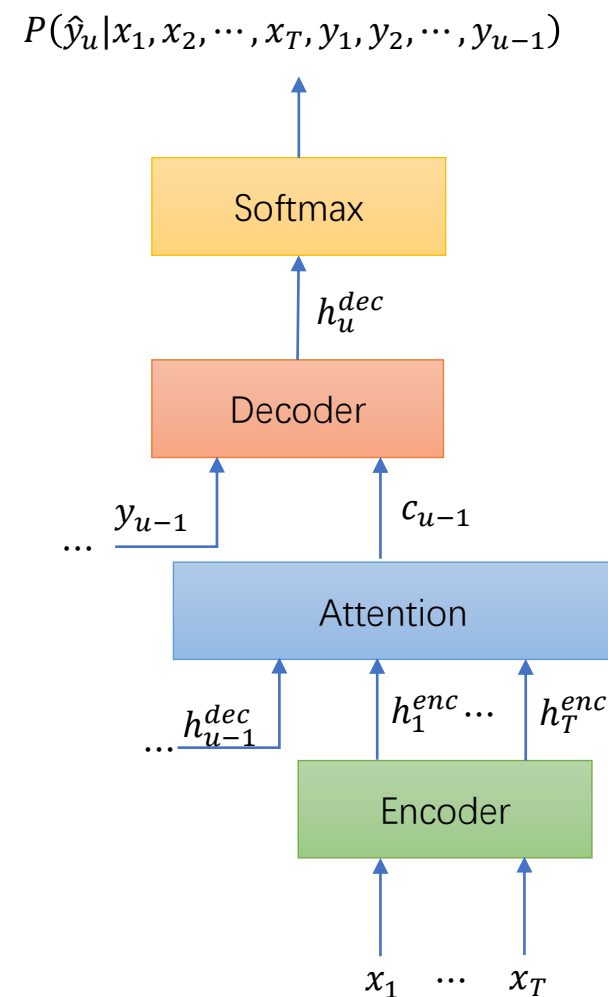
# EspNet中的声学模型

Attention

```
# network architecture
# encoder related
etype: vggblstm # encoder architecture type
elayers: 3
eunits: 1024
eprojs: 1024
subsample: "1_2_2_1_1" # skip every n frame from input to nth
layers
# decoder related
dlayers: 2
dunits: 1024
```

```
.....
# attention related
atype: location
adim: 1024
aconv-chans: 10
aconv-filts: 100
```

```
.....
# hybrid CTC/attention
mtlalpha: 0.5
```



# EspNet中的声学模型

Transformer

# network architecture

# encoder related

elayers: 12

eunits: 2048

# decoder related

dlayers: 6

dunits: 2048

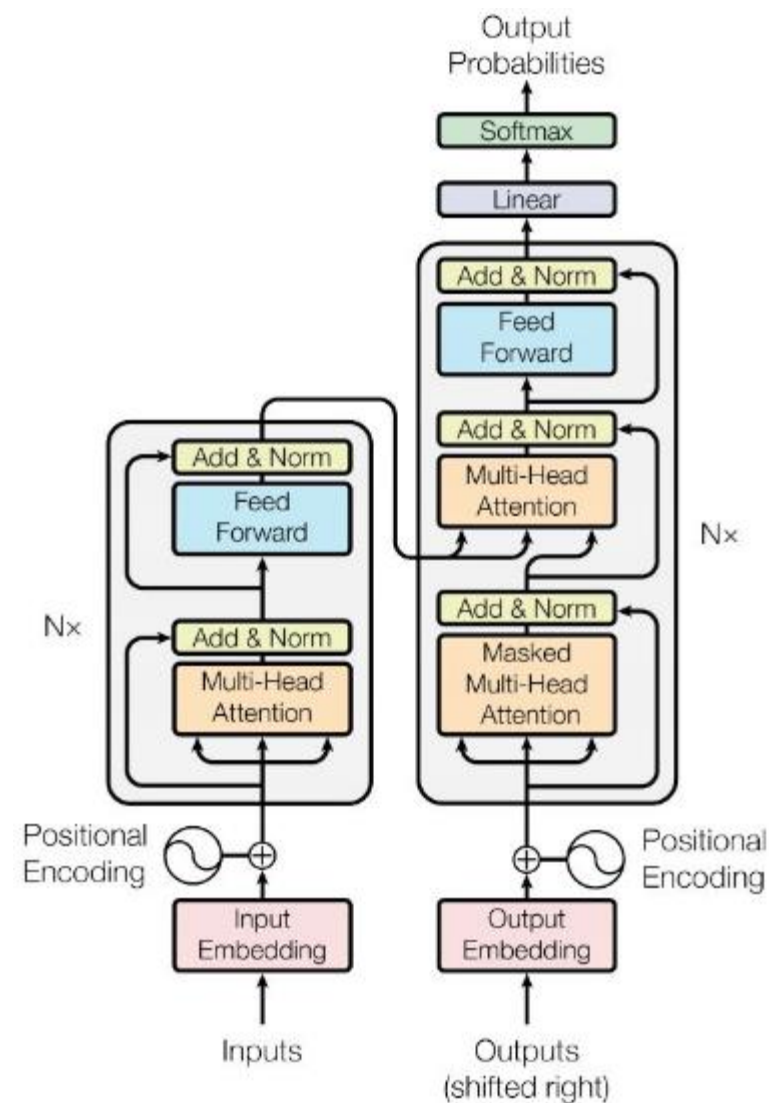
# attention related

adim: 256

aheads: 4

.....  
# hybrid CTC/attention

mtlalpha: 0.3

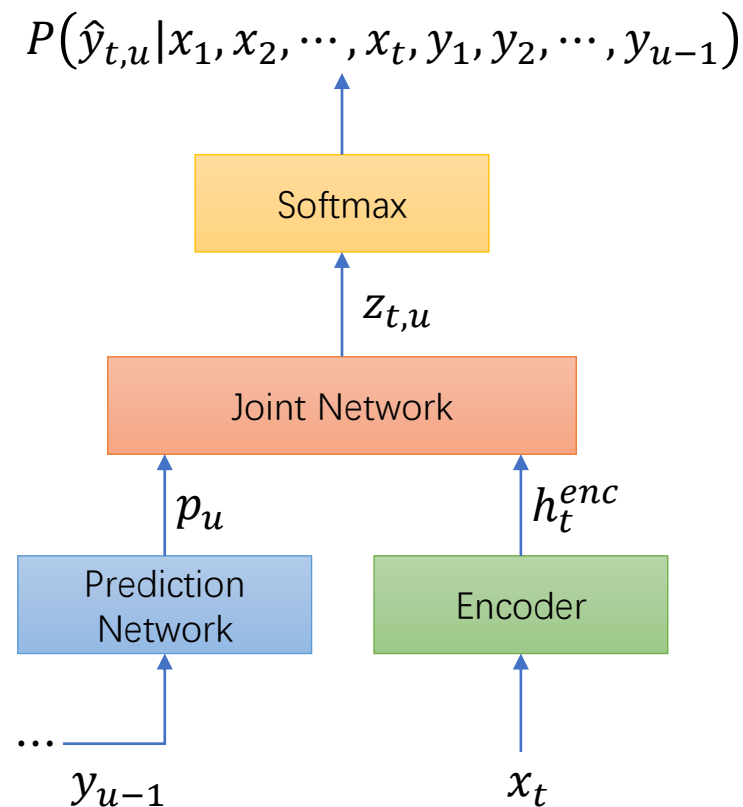


图片来源: "Attention Is All you Need"

# EspNet中的声学模型

RNN-T

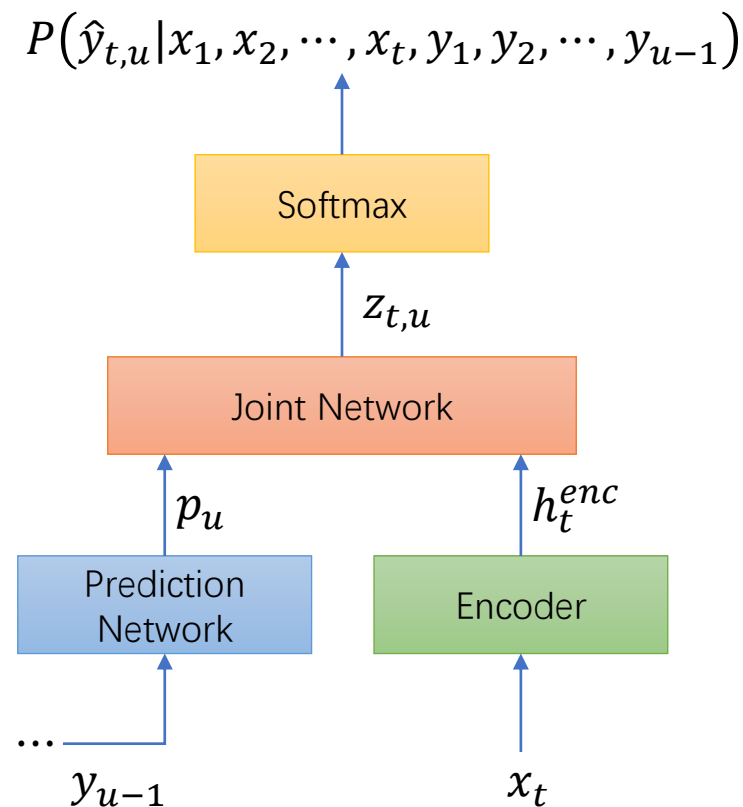
# network architecture  
## encoder related  
etype: blstm  
elayers: 4  
eunits: 320  
eprojs: 320  
subsample: "1\_2\_2\_1\_1"  
dropout-rate: 0.0  
## decoder related  
dtype: lstm  
dlayers: 1  
dec-embed-dim: 300  
dunits: 300  
## joint network related  
joint-dim: 300



# EspNet中的声学模型

Transformer-T

# network architecture  
## encoder related  
etype: transformer  
transformer-input-layer: vgg2l  
elayers: 8  
eunits: 320  
dropout-rate: 0.4  
## decoder related  
dtype: transformer/lstm  
dlayers: 2  
dec-embed-dim: 300  
dunits: 300  
dropout-rate-decoder: 0.1  
## attention related  
adim: 320  
aheads: 4  
## joint network related  
joint-dim: 300



# 训练过程

## 1. Espnet配置文件

a) 训练Transformer网络模型的配置文件 train.yaml

```
# network architecture
# encoder related
elayers: 4
eunits: 1024
# decoder related
dlayers: 4
dunits: 1024
# attention related
adim: 128
aheads: 4

# hybrid CTC/attention
mtlalpha: 0.3
```



# 训练过程

## b) 语言模型的训练配置文件 lm.yaml

```
# rnnlm related
layer: 2
unit: 650
opt: sgd      # or adam
batchsize: 32 # batch size in LM training
epoch: 20     # if the data size is large, we can reduce this
patience: 3
maxlen: 100
```

# 训练过程

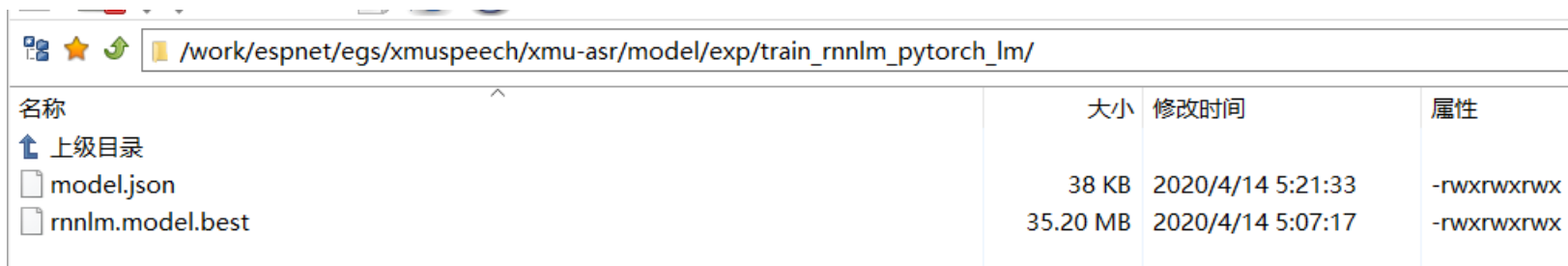
c) 解码的配置文件 decode.yaml

```
batchsize: 0  
beam-size: 20  
penalty: 0.0  
maxlenratio: 0.0  
minlenratio: 0.0  
ctc-weight: 0.5  
lm-weight: 0.7
```

# 训练过程

## 2. 语言模型训练

使用lm\_train.py 进行RNNLM语言模型的训练，会得到每个epoch训练好的模型，并挑选了一个最优的模型，将其命名为rnnlm.model.best。



名称	大小	修改时间	属性
上级目录			
model.json	38 KB	2020/4/14 5:21:33	-rwxrwxrwx
rnnlm.model.best	35.20 MB	2020/4/14 5:07:17	-rwxrwxrwx

# 训练过程

## 3. 声学模型训练

这一步通过shell脚本调用Espnet中的声学模型训练脚本asr\_train.py实现，在声学模型训练过程中，需要准备4个文件：

- a) 声学模型训练配置文件
- b) 词典文件
- c) 对应于训练集数据的 data.json 文件
- d) 对应于验证集数据的 data.json 文件

# 测试结果

EspNet 的语音识别解码器使用脚本 `asr_recog.py` 来实现，默认使用 CPU 进行解码，如果要使用 GPU 进行解码，需要在 `decode` 配置文件中添加 `api=v2`。解码过程需要使用到前面准备和训练得到的几个文件：

- a) `cmvn.ark`
- b) `rnnlm.model.best`
- c) `model.acc.best`

# 测试结果

在解码文件输出目录中，会生成一个 result.txt 文件，该文件保存了最终在测试集和验证集上的解码结果，如下图所示：

exp/train_sp_pytorch_train/decode_test_decode_lm/hyp.trn									
SPKR	# Snt	# Wrd	Corr	Sub	Del	Ins	Err	S.Err	
s001	350	3833	97.5	1.9	0.5	3.0	5.5	20.6	
s002	392	4368	96.1	2.5	1.4	1.3	5.2	16.1	
s003	350	3813	95.6	3.4	1.0	3.4	7.8	27.1	
Sum/Avg	1092	12014	96.4	2.6	1.0	2.5	6.1	21.1	
Mean	364.0	4004.7	96.4	2.6	1.0	2.6	6.1	21.3	
S.D.	24.2	314.8	1.0	0.8	0.4	1.1	1.4	5.6	
Median	350.0	3833.0	96.1	2.5	1.0	3.0	5.5	20.6	

# 测试结果

在result.txt中，还可以查看一条语音对应的预测句子，如下图所示：

```
Speaker sentences 0: s001 #utts: 350
id: (s001-record2020s001w001)
Scores: (#C #S #D #I) 11 0 0 0
REF: 会 注 意 起 跑 时 谁 更 快 一 些
HYP: 会 注 意 起 跑 时 谁 更 快 一 些
Eval:

id: (s001-record2020s001w002)
Scores: (#C #S #D #I) 16 0 0 0
REF: 人 这 一 辈 子 遇 见 对 你 好 的 人 比 较 容 易
HYP: 人 这 一 辈 子 遇 见 对 你 好 的 人 比 较 容 易
Eval:

id: (s001-record2020s001w003)
Scores: (#C #S #D #I) 15 0 0 0
REF: 如 果 这 样 的 要 求 都 没 人 能 接 受 的 话
HYP: 如 果 这 样 的 要 求 都 没 人 能 接 受 的 话
Eval:

id: (s001-record2020s001w004)
Scores: (#C #S #D #I) 8 0 0 0
REF: 这 根 本 就 是 抢 钱 啊
HYP: 这 根 本 就 是 抢 钱 啊
Eval:
```

# EspNet中的声学模型性能

在不同数据上各种模型的平均性能对比：

1.性能：  $\text{Transformer+CTC} > \text{Transformer} \geq \text{Attention+CTC} > \text{Attention} \geq (\text{Transformer-T}) > \text{RNN-T} > \text{CTC}$

2.实时性：  $\text{Attention+CTC} < \text{Attention} < \text{Transformer+CTC} < \text{Transformer} (\text{Transformer-T}) \leq \text{RNN-T} < \text{CTC}$



# Espnet中的声学模型性能

实验结果补充：Transformer/CTC、Attention/CTC模型与Kaldi实验结果在dev和test数据集上的对比。

官方：

dataset	token	error	Kaldi (s5)	ESPnet RNN (ours)	ESPnet Transformer (ours)
AISHELL	char	CER	N/A / 7.4	6.8 / 8.0	<b>6.0 / 6.7</b>
AURORA4	char	WER	(*) 3.6 / 7.7 / 10.0 / 22.3	3.5 / 6.4 / 5.1 / 12.3	<b>3.3 / 6.0 / 4.5 / 10.6</b>
CSJ	char	CER	(*) 7.5 / 6.3 / 6.9	6.6 / 4.8 / 5.0	<b>5.7 / 4.1 / 4.5</b>
CHiME4	char	WER	<b>6.8 / 5.6 / 12.1 / 11.4</b>	9.5 / 8.9 / 18.3 / 16.6	9.6 / 8.2 / 15.7 / 14.5
CHiME5	char	WER	<b>47.9 / 81.3</b>	59.3 / 88.1	60.2 / 87.1
Fisher-CALLHOME Spanish	char	WER	N/A	27.9 / 27.8 / 25.4 / 47.2 / 47.9	<b>27.0 / 26.3 / 24.4 / 45.3 / 46.2</b>
HKUST	char	CER	23.7	27.4	<b>23.5</b>
JSUT	char	CER	N/A	20.6	<b>18.7</b>
LibriSpeech	BPE	WER	3.9 / 10.4 / 4.3 / 10.8	3.1 / 9.9 / 3.3 / 10.8	<b>2.2 / 5.6 / 2.6 / 5.7</b>
REVERB	char	WER	18.2 / 19.9	24.1 / 27.2	<b>15.5 / 19.0</b>
SWITCHBOARD	BPE	WER	<b>18.1 / 8.8</b>	28.5 / 15.6	<b>18.1 / 9.0</b>
TED-LIUM2	BPE	WER	<b>9.0 / 9.0</b>	11.2 / 11.0	9.3 / <b>8.1</b>
TED-LIUM3	BPE	WER	<b>6.2 / 6.8</b>	14.3 / 15.0	9.7 / 8.0
VoxForge	char	CER	N/A	12.9 / 12.6	<b>9.4 / 9.1</b>
WSJ	char	WER	<b>4.3 / 2.3</b>	7.0 / 4.7	6.8 / 4.4

我们的实验结果(dev/test):

dataset	token	error	kaldi(our)	ESPnet RNN(our)	ESPnet Transformer(our)	ESPnet Transformer+ctc(our)
AISHELL-1	char	CER	N/A / 7.51	6.8/8.1	6.7/7.9	6.0/6.7

对比结果可以发现，和官方提供的差别不大。同学们可以根据前面所讲的内容，进行其他模型的实践。

# 致谢

- 感谢李松同学对Espnet实践过程做了深入细致的整理。
- 感谢厦门大学智能语音实验室其他同学的贡献。