# 孤立词语音识别项目报告

16307130194 陈中钰

# 1. 背景

## 1.1项目概要

- 给定 20 个孤立词语的录音文件，训练出一个孤立词分类模型。

## 1.2语音特征提取

- MFCC 梅尔倒谱系数
- 语音因素
- 语谱图

## 1.3语音识别

- 分类神经网络
- LBG 聚类方法
- HMM 隐马尔可夫模型
- DTW 动态时间规划模式匹配

# 2. MFCC 特征提取

## 2.1 代码组织

| 文件名 | 功能 |
| --- | --- |
| utils.py | 定义了语音文件的检查和读入的函数，并对数据集进行划分。 |
| spectrogram.py | 处理语音并生成语谱图，用 pickle 储存数据集。 |
| mfcc.py | 处理语音并生成 MFCC，用 pickle 储存数据集。 |
| config.ini | 定义了上述数据处理、特征提取过程中所用到的全部参数。 |

## 2.2 端点检测

### 2.2.1 分帧

把语音分割为固定长度的帧，从每一帧获取对应的短时能量、短时过零率的局部值，进而能在全局时间范围内反映语音的短时能量、短时过零率的变化。帧长一般为 10~30ms，帧移一般为 0~0.5 倍的帧长。在这里，帧长 $frame\_size = 0.03s$，帧移是帧长一半，即$frame_stride = 0.015s$。

此外，还需要通过乘以采样率$\mathrm{round}(\mathrm{time} * \mathrm{sample_rate})$，把时间域上的帧长和帧移转化为离散信号上的帧长 480 和帧移 240。此外为了保证每帧长度一致，需要把语音信号在末尾补 0，使信号长度为帧长的整数倍：$\mathrm{k} * \mathrm{frame_length}, \mathrm{k} \in \mathrm{N}$。通过 numpy 可以容易把补整了的语音信号划分为若干帧。

由于语音信号为 2s，也就是 32000 个数据点，最终划分为 132 帧，形状为 $(132, 480)$。

### 2.2.2 加窗

对语音信号的每一帧$\mathrm{s(n)}$加上一个低通滤波器$\mathrm{w(n)}$，可以增强低频成分（语音）、降低高频成分（噪音），使获得的特征更有代表性、更不易受噪声影响。在课上学习到的 4 种窗中，汉明窗具有最大的通带外衰减，低通的效果最好，因此选择汉明窗。

$$\mathrm{s}_w(\mathrm{n}) = \mathrm{s(n)w}_{hamming}(\mathrm{n})$$

### 2.2.3 短时能量

浊音的短时平均能量比清音更高，而本次录音数据集的噪声一般都很小，因此噪声短时能量很低，所以通过短时平均能量可以找出语音的浊音部分。通过以下公式可以计算出语音信号每一帧的能量$\mathrm{E}$。

$$\mathrm{E} = \sum_n \mathrm{s}_w(n)$$

通过把信号短时能量序列进行归一化，并设置短时能量下界为 0.2，可以获得浊音的起始范围，可以作为初步的端点检测结果。

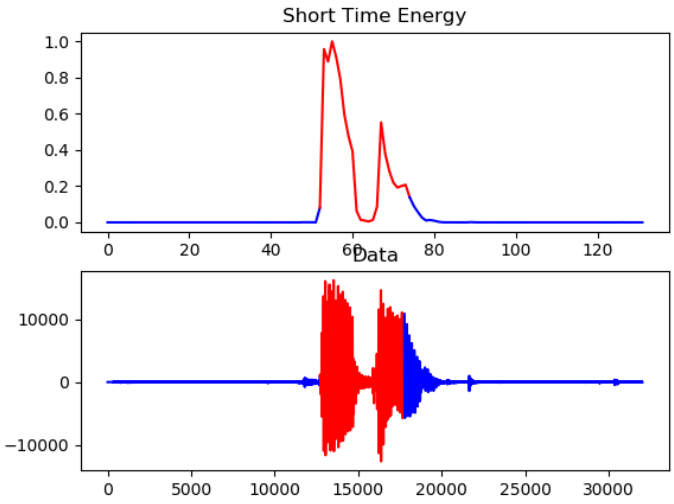


图 1 语音 16307130194-10-10.wav 的短时能量检测结果

### 2.2.4 短时过零率

浊音的短时过零率约为 14 次/10ms，清音的短时过零率约为 47 次/10ms，而由于前后相关性极低，噪音的短时过零率更高。由于帧长为 15ms，因此取短时过零率上界为 80 次/15ms，可以把清音、浊音和噪音分离。此外，也可以利

用短时过零率把清音和浊音分离，用于音素的提取，构造 VQ 码本进行语音的识别。

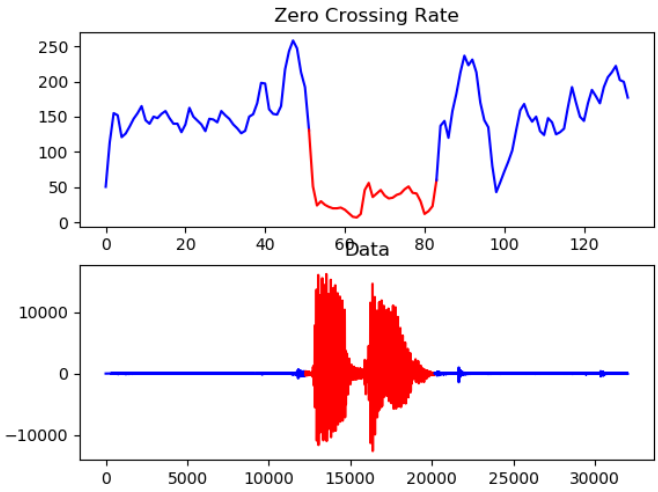


图 2 语音 16307130194-10-10.wav 的短时过零率检测结果

### 2.2.5 联合检测端点

上述两种短时参数各有优略，因此可以联合两者进行端点检测。本次实验采取的策略为，先通过短时能量划分出初步的起始点范围，接着了利用短时过零率来扩展该范围，获得最终的起始点范围。最后把语音起始点以外的信号置零，去除无关噪声，并把语音信号放到信号中央，尽量消除语音所在位置对后续识别的影响。

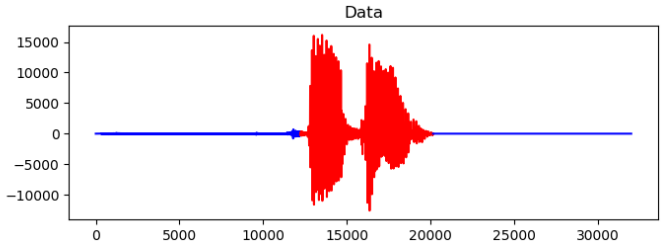


图 3 语音 16307130194-10-10.wav 的端点检测结果

## 2.3 预加重

在正式求解 MFCC 系数前，可以给信号加上一个一阶滤波器，用于简单地提高信号高频成分的强度，使频谱里的高频、低频成分的强度更平衡。此外，预加重还能提高信噪比。这个滤波器可以通过以下公式实现，其中选取预加重系数α为 0.97。

$$y(t) = x(t) - \alpha * x(t-1)$$

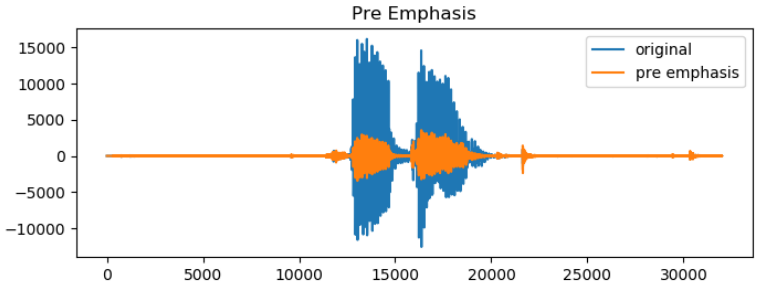


图 4 语音 16307130194-10-10.wav 的预加重结果

## 2.4 短时傅里叶变换 STFT

### 2.4.1 准备

在求解 STFT 之前,同样也需要对信号进行分帧、加窗的操作，这两步操作和上文叙述的一样，相关的参数也保持一致。

### 2.4.2 快速傅里叶变换 FFT

STFT 的实现是基于 DFT 的，那么首先要实现 DFT。在信号为以 2 为基的情况下，可以利用蝶形变换的性质来实现 FFT，使运算速度大大提高。

$$\begin{cases} \mathrm{X(k)} = \sum_{n=0}^{N-1} x(n)W_N^{nk} = \sum_{r=0}^{N/2-1} e(r)W_{N/2}^{rk} + W_N^{nk} \sum_{r=0}^{N/2-1} o(r)W_{N/2}^{rk} = E(k) + W_N^k O(k) \\ \mathrm{X}\left(\mathrm{k} + \frac{N}{2}\right) = E(k) - W_N^k O(k) \end{cases}$$

通过把信号中的奇数、偶数点序列分开，可以获得 DFT 的递归公式，由奇数点序列的 DFT 变换和偶数点序列的 DFT 变换以及旋转因子求得整个序列的 DFT 变换：

$$\mathrm{X(k)} = [E(k) + W_N^k O(k), E(k) - W_N^k O(k)\,]$$

其中$W_N^k = \exp\left(-\frac{2jk\pi}{N}\right)$

由于信号以 2 为基，因此可以不断把 FFT 求解划分为相同大小的两部分的 FFT 求解，直到序列只有一个元素，那么直接作为结果返回即可。

分析可得，DFT 复杂度为$\mathrm{O(n^2)}$，而由于 FFT 每次求解都可以递归为信号奇偶两半的 FFT 变换，故复杂度为$\mathrm{O(nlogn)}$。

通过 python 的切片可以容易获得奇序列和偶序列，通过递归求解，最后把两部分结果通过 hstack 拼在一起即可。由于分帧后，每一帧的大小为 480，而不是 512，因此在进行 FFT 递归计算前，需必须要在每一帧的末尾补 0 补成长度为 512 每帧。

### 2.4.3 短时傅里叶变换 STFT

短时傅里叶变换就是对每一帧进行 FFT，并最后取绝对值，保留强度。短时傅里叶变换反映了语音信号的频谱随时间的变化，具有时间、频率两个维度。

### 2.4.4 语谱图能量谱

通过以下公式可以计算语谱图的能量谱。此外，这个能量谱也具有很好的语音特征，因此在本次实验中，也用作为语音识别的输入特征数据。

$$\mathrm{P} = \frac{|STFT(n)|^2}{N}$$

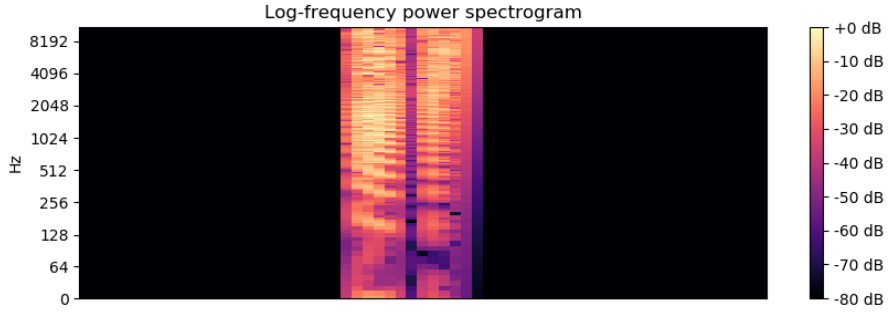


图 5 语音 16307130194-10-10.wav 的对数语谱图强度

### 2.4.5 Mel 滤波器组

人耳对声音频率的响应不是线性的，对低频信号比对高频信号更敏感，但是对 Mel 对数响应$mel = 2595log_{10}(1 + f/700)$是线性的，为了模拟这种人耳的特性，用 Mel 滤波器组对信号进行处理，提取出频率带。

Mel 滤波器组由一组三角形滤波器组成，在 Mel 频率域中，三角形滤波器的宽度是相等的，三角形峰值都为 1，相邻两个滤波器重叠一半。

对信号采用 40 个滤波器的 Mel 滤波器组，首先把语音信号频率范围映射到 Mel 频率域中，从中产生均匀分布的 40 个三角形滤波器峰值对应的 Mel 频率，通过 Mel 反变换$f = 700(10^{mel/2595} - 1)$转换回频率域中，通过以下公式产生第 m 个三角形滤波器：

$$\mathrm{H}_m(k) = \begin{cases} 0, k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)}, f(m-1) \le k < f(m) \\ 1, k = f(m) \\ \dfrac{f(m+1-1)}{f(m+1) - f(m)}, f(m) < k \le f(m+1) \\ 0, > f(m+1) \end{cases} \qquad m = 1,2,\ldots,40$$

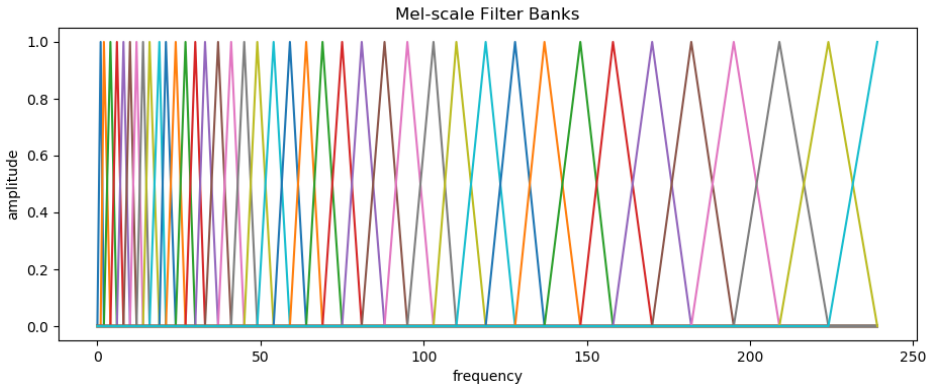


图 6 Mel 滤波器组在频率域中的分布

生成滤波器组后，使信号通过滤波器组只需要与滤波器组矩阵相乘。另外，需要把数值为 0 的部分加上一个很小的值，使得后续可以进行对数计算。

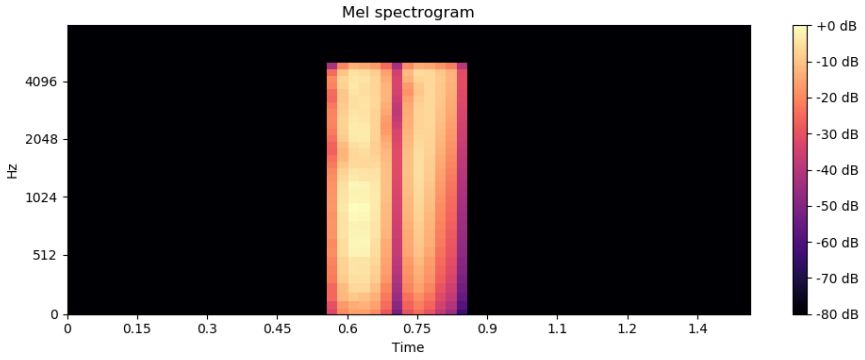


**图 7 语音 16307130194-10-10.wav 的 Mel 频谱**

### 2.4.6 对数操作

对上述获得的 Mel 频谱进行对数操作。

### 2.4.7 离散余弦变换 DCT

本次实验中，离散余弦变换基于以下 DCT-II 公式实现。实验中设置了 12 个 MFCC 系数，因此最终结果取 DCT 变换结果的 1～12 列，即为 MFCC 系数

$$\mathrm{X}(k) = \sum_{n=0}^{N-1} x(n)cos(\frac{\pi}{N}(n+1/2)k), k = 0,1,2\ldots,N-1$$

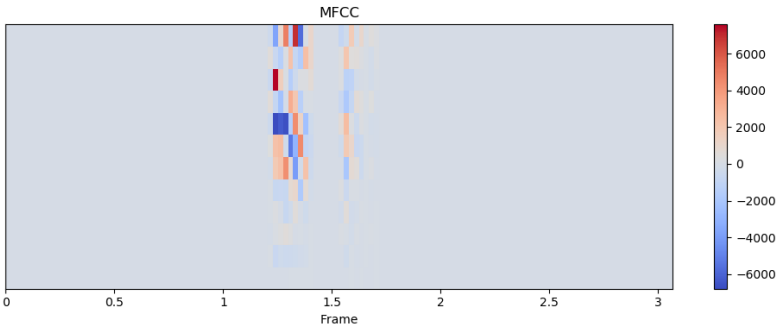


**图 8 语音 16307130194-10-10.wav 的 MFCC**

### 2.4.8 Delta MFCC

MFCC 系数只能反映单个帧的特征，不能体现帧之间的相关特征。而语音信号有很强的前后相关性，因此需要求出 Delta MFCC 来提取不同帧之间的相关特征。

通过以下公式，可以求出不同帧之间的相关的特征，每一帧的系数也同样是 12 个，和上述 MFCC 特征拼接在一起就可以获得最终的 MFCC 系数，长度为 24。

$$\mathrm{deltaMFCC} = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

本次实验中，最后没有实现这一步，只使用了原始的 MFCC 特征。

# 3. 卷积神经网络识别模型

（代码可见 cnn.py）

## 3.1 简单的 CNN

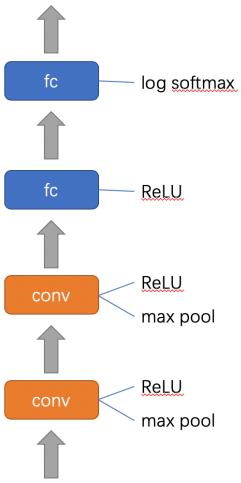


图 9 CNN 结构

## 3.2 VGG11 with batch normalization

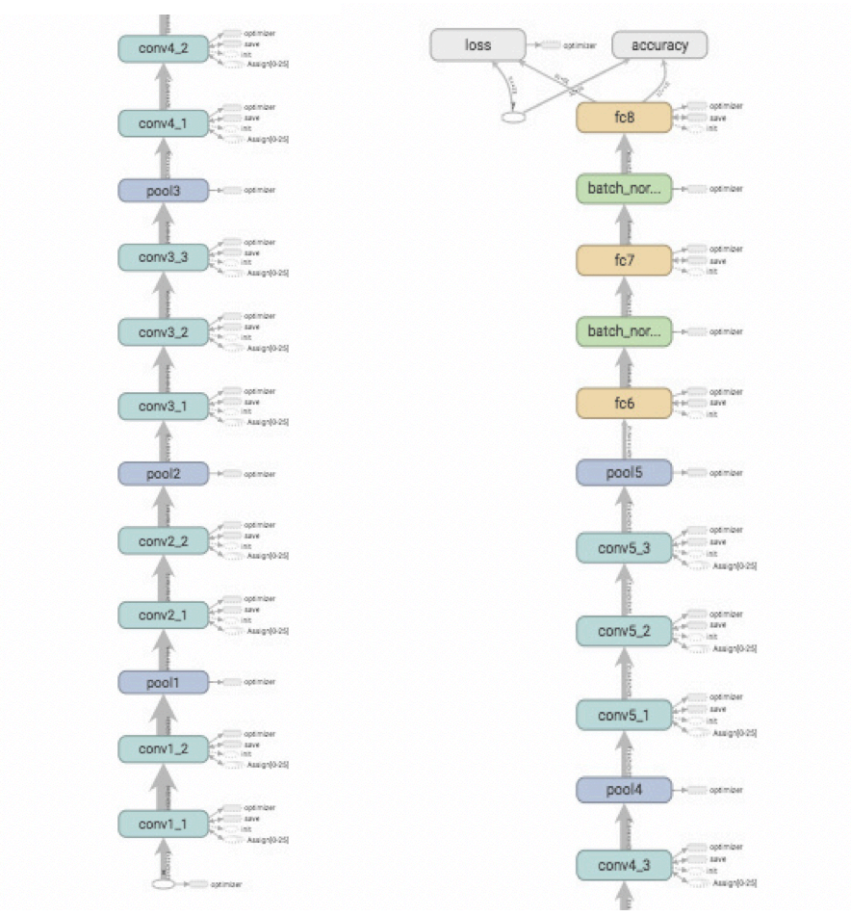


图 10 VGG 结构

# 4. 实验设置

## 4.1 代码结构

| 文件名 | 功能 |
| --- | --- |
| config.ini | 定义了上述数据处理、特征提取、模型训练中所用到的全部参数。 |
| cnn.py | 定义模型类。 |
| run_cnn.py | 定义了模型训练的 train、test 等函数。 |
| dataset.py | 定义了自己的 torch 的 dataset 类。 |
| server.py | 定义 flask 的后端代码。 |

## 4.2 特征提取

### 4.2.1 数据清洗

用文件名格式进行筛选，忽略掉一些隐藏文件和一份命名错误的音频文件。最终获得 13599 份有效语音文件

### 4.2.2 数据集划分

由于老师的声音是测试集，是完全不在训练集和开发集里的，因此对 34 个录音的同学进行 20%为开发集的划分，最终得到训练集有 11599 份语音文件、开发集有 2000 份语音文件。

#### 4.2.3 特征获取

如上文所述 MFCC 特征提取过程所述，提取出语谱图的能量谱、MFCC 系数作为训练特征，分别用于模型的训练，并对比两个特征集的训练效果。

#### 4.2.4 模型输入

获取特征数据后，导入到符合 torch 要求的 DataSet 类中，并导入到 torch 的 DataLoader 中。

### 4.3 参数设置

| 参数 | 值 |
|---|---|
| batch size | 64 |
| max epoch | 32 |
| learning rate | 0.001 |
| optimizer | Adam |
| loss | Cross Entropy Loss |
| metric | Accuracy |

## 5. 实验结果与分析

### 5.1 实验结果

| 模型 | CNN | | VGG | |
|---|---|---|---|---|
| 特征输入 | 语谱图能量谱 | MFCC | 语谱图能量谱 | MFCC |
| 开发集准确率 | 74.5 | 82.25 | 83.75 | 94.4 |
| best epoch | 24 | 15 | 29 | 28 |
| 训练用时 | 10～20min | | ～1h | |

### 5.2 实验结果图表

#### 5.2.1 loss

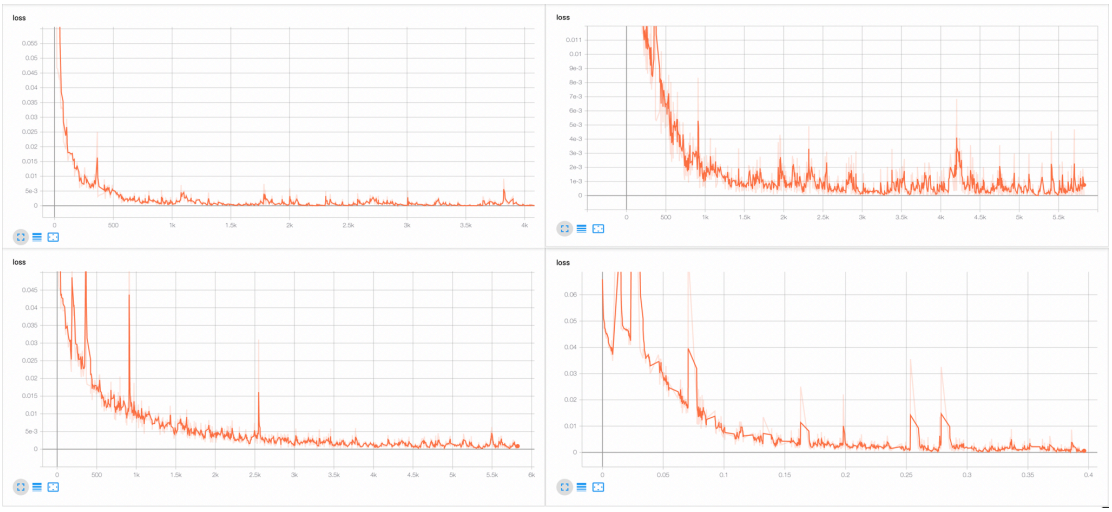

图 11 左上方为 CNN+语谱图，右上方为 CNN+MFCC，
左下方为 VGG+语谱图，右下方为 VGG+MFCC

### 5.2.2 accuracy

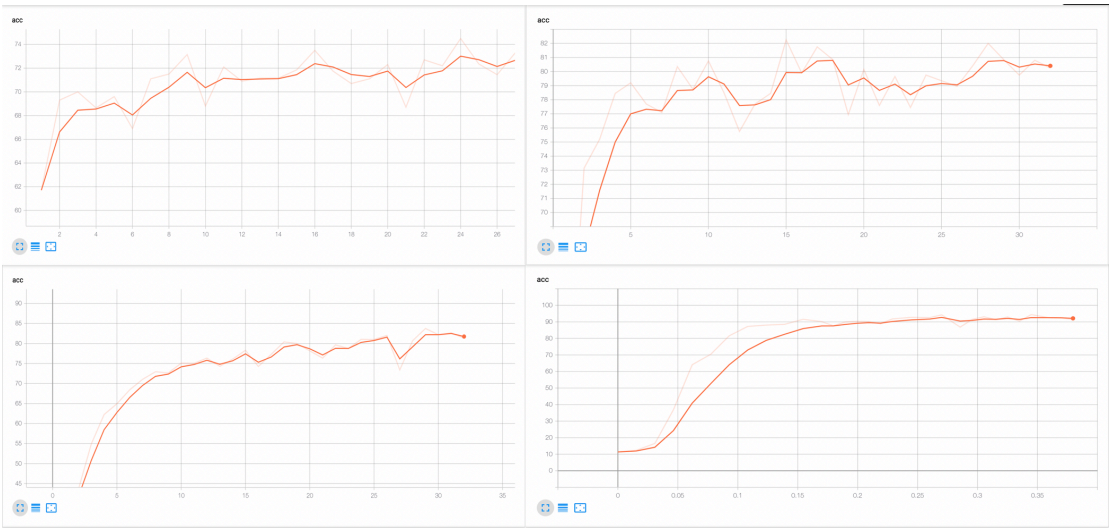

图 12 左上方为 CNN+语谱图，右上方为 CNN+MFCC，
左下方为 VGG+语谱图，右下方为 VGG+MFCC

# 6. 结论

## 6.1 整体结果

总体上来说，准确率可以达到约 94%，可以很好地做到给定孤立词的分类。

## 6.2 模型对比

经过对比可以发现，深层的分类网络比浅层的分类网络效果都更好，能提高～10%，不过训练所需要的时间也大幅上升。

对比语谱图能量谱和 MFCC 作为训练特征，MFCC 的结果都更好，可以认为 MFCC 能够包含更多的语音特征，更适合用在本次实验的语音集上。

# 7. 参考文献

[1] Simonyan K , Zisserman A . Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.

# 8. 附件

本次课程的网页录音工具已上传到 https://github.com/czhongyu/audio-collector，在最后附上录音工具的使用文档。

另外我还收集、整理了同学的录音文件，在 https://github.com/czhongyu/DSPSpeech-20，在这里也附上数据的说明文档。