

# 《视听信息系统导论》第三次大作业：

## 基于音频和图像序列的物体撞击匹配

A L<sup>1</sup>, 姜安柏<sup>2</sup>, Xt X<sup>3</sup>

### 1 整体完成情况

本次大作业完成了全部三个任务，输出接口已适配到 test.py 中，全部测试集可以运行。

任务一采用神经网络进行音频分类；任务二和任务三均先采用神经网络分别对音频和视频分类，每个类内再进行匹配。

训练集上训练时，音频分类网络准确率超过 93%，视频分类网络的准确率接近 100%。

### 2 原理

#### 2.1 任务一

任务一需要对测试集的音频进行 10 分类，即给出各个音频对应的物体种类。

首先，我们采用梅尔频率倒谱系数 MFCC 作为声音的特征。MFCC 是基于人耳听觉特性提出的频谱特征，可以准确描述短时音频功率谱的包络特征。我们采用 40 维 MFCC 结果的时间平均值作为单个通道的特征。4 个通道的特征合在一起，作为整个音频的特征。

然后，我们将训练集的音频特征送入卷积神经网络进行训练。训练好后再提取测试集的音频特征，送入神经网络进行分类

#### 2.2 任务二、任务三

任务二、任务三均需要寻找音频和视频的对应关系。我们采用分类-提取特征-匹配的思路进行处理。

##### 2.2.1 分类

我们对音频和视频分别进行分类，并将同类的音频和视频的索引放在一起。

其中音频分类沿用任务一的思路；视频分类使用 RGB 图像训练神经网络。分类结果如下：

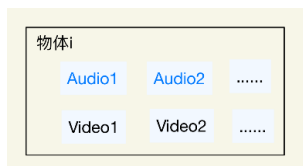


图 1 分类结果示意图

##### 2.2.2 匹配特征提取

我们对各段音频和视频分别提取 2 类相同的特征，然后利用这些特征进行匹配。选择提取的特征如下：

	音频	视频
强度特征	最大功率	运动速度
碰撞特征	碰撞边	碰撞边

选取强度特征是出于一个基本的假设：相同物体，运动快的物体碰撞声音大，碰撞声音大的物体运动快。对于音频，我们提取各声道最大瞬时功率作为强度特征；对于视频，我们找到物体的中心，计算物体中心的帧间位移，提取最大帧间位移作为视频的强度特征。两个特征分别进行归一化。

选取碰撞边是因为碰撞边的识别更为鲁棒。音频上，碰撞时刻相应边的音频信号会出现一个很大的阶跃，易于判断。视频上，我们通过物体边缘与托盘边的接触情况判断碰撞边，此部分判断存在一定误差，因而我们根据物体的运动速度计算出给出碰撞边的置信系数（0 或 1）以增加匹配的准确性。

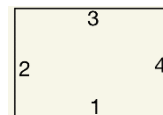


图 2 碰撞边与音频 4 通道的对应关系

##### 2.2.3 匹配算法

任务二和任务三采用相同的分类算法和特征提取算法，但在匹配算法的选择上则有所不同。

任务二是完全匹配，因而我们选择遍历同类物体的所有音频视频的最大组合方式，计算各组合方式的误差，寻找误差最小的组合方式。其中最大组合方式是指所有音频和视频都要参与匹配，仅当音频和视频量不等的时候才会出现未匹配项。在各个类别的匹配都完成后，我们再对全部未匹配项进行匹配，以保证最终所有音频和视频都有所对应。

任务三是不完全匹配，因而我们选择贪心算法，每次只提取误差最小的组合，当最

小误差大于某个阈值后，我们认为匹配完成，剩下的都是未匹配的。

为描述误差大小，我们定义了一种损失函数，对同类特征的差值按照其差值的大小进行加权。针对强度特征，我们定义了分区间加权函数；针对碰撞特征，我们根据视频碰撞边的置信系数进行加权。最终损失函数为 2 个特征各自损失函数之和。

### 3 具体实现

#### 3.1 任务一

我们先根据来自托盘四条边中央位置的麦克风音频计算 MFCC，每个通道的结果按照时间求均值，得到 4 个 40 维向量，这 4 个向量合在一起，作为该音频的特征，即每段音频提取出 160 个数据点。

然后，我们将这 160 个数据点送入卷积神经网络训练。音频分类网络结构如下：

```
net = nn.Sequential(
    (0): Conv2d(4, 16, kernel_size=(1, 1), stride=(1, 1), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=1, stride=1, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
)
classifier = nn.Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=640, out_features=2240, bias=True)
)
net = nn.Sequential(net, classifier)
```

图 3 音频分类网络

测试时提取音频的 MFCC，直接送入训练好的神经网络，输出结果。

#### 3.2 任务二

首先用卷积神经网络对音视频分类，将属于同类物体的音视频放在一个字典的 key 下面。神经网络结构如下：

```
net = nn.Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=1, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
)
classifier = nn.Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=288, out_features=18, bias=True)
)
net = nn.Sequential(net, classifier)
```

图 4 图像分类网络

然后分别提取音频和视频中用于匹配的特征。对于音频，提取出最大的瞬时功率  $P_m$  和相应的声道  $i$ ，例如

$$Audio1 = [P_m, i]$$

对于视频，先计算出物体的轮廓，然后

求出重心，对相邻帧的重心坐标作差得到速度，取出最大的速度  $v_m$  和对应的边  $j$ ，由于视频相应特征提取难度较大，这里引入置信因子  $p$ ，例如

$$Video1 = [v_m, j, p]$$

在每个物体对应的 key 下面，先将  $P_m$  与  $v_m$  归一化，随后计算每一对音视频两个特征的损失函数。对于两种误差给予不同的加权系数，其中衡量“撞击剧烈程度”的特征采用动态加权系数，一对音视频特征相差越大，则此加权系数越大；判定碰撞边的特征权重设置为  $\max(p, threshold)$ 。则每一对音视频的误差定义为

$$e = |P_m - v_m| * weight0 + |i - j| * weight1$$

对于同类物体的匹配，认定所有可能的音频-视频全匹配中，总误差最小的匹配为最终匹配结果。

由于神经网络分类时不能保证全部分对，我们在各个类内匹配都完成后再对所有没有匹配上的音频和视频进行匹配，匹配算法同类内匹配。

#### 3.3 任务三

与任务二类似，由于这里存在没有对应的音频和视频，因此使用贪心算法进行匹配。特征提取的步骤与任务二相同，匹配时，先选出两个误差之和最小的一对音视频，然后在剩下的音视频中重复上述操作。同时设定一个阈值，对于误差超过阈值的音视频，认为其不存在对应的音频或视频。

## 4 结果展示

#### 4.1 任务一

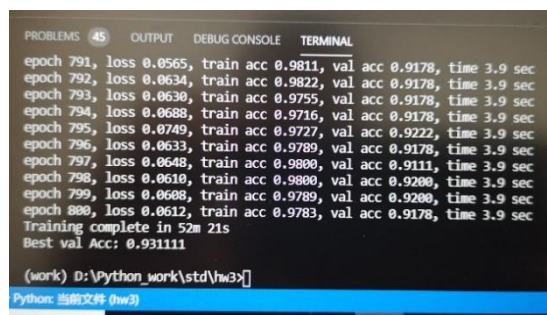


图 5 音频识别训练测试

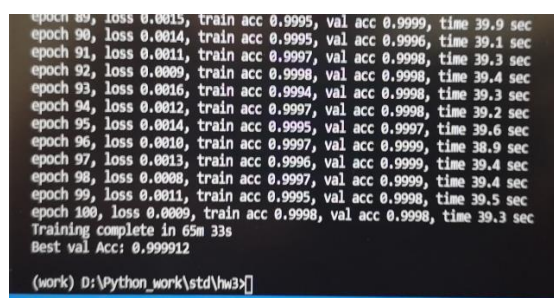
训练时，我们将所有训练集按照 8:2 的比例切分成训练集和验证集。训练集准

准确率接近 98%，验证集的准确率达到 93.11%。

## 4.2 任务二

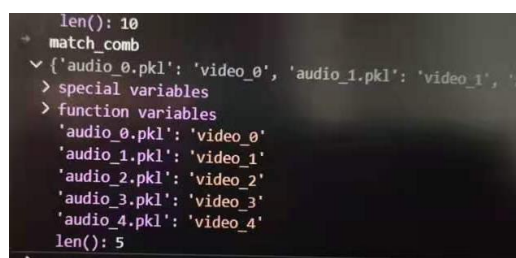
我们先对图像分类进行了训练，正确率(*Best val Acc*)几乎达到 100%

随后，我们从训练集中选取了 5 对音频和对应的视频进行匹配测试，结果如下：



```
epoch 89, loss 0.0015, train acc 0.9995, val acc 0.9999, time 39.9 sec
epoch 90, loss 0.0014, train acc 0.9995, val acc 0.9996, time 39.1 sec
epoch 91, loss 0.0011, train acc 0.9997, val acc 0.9998, time 39.3 sec
epoch 92, loss 0.0009, train acc 0.9998, val acc 0.9998, time 39.4 sec
epoch 93, loss 0.0016, train acc 0.9994, val acc 0.9998, time 39.3 sec
epoch 94, loss 0.0012, train acc 0.9997, val acc 0.9998, time 39.2 sec
epoch 95, loss 0.0014, train acc 0.9995, val acc 0.9997, time 39.6 sec
epoch 96, loss 0.0010, train acc 0.9997, val acc 0.9999, time 38.9 sec
epoch 97, loss 0.0013, train acc 0.9996, val acc 0.9999, time 39.4 sec
epoch 98, loss 0.0008, train acc 0.9997, val acc 0.9999, time 39.4 sec
epoch 99, loss 0.0011, train acc 0.9995, val acc 0.9998, time 39.5 sec
epoch 100, loss 0.0009, train acc 0.9998, val acc 0.9998, time 39.3 sec
Training complete in 65m 33s
Best val Acc: 0.999912
(work) D:\Python_work\std\hw3>
```

图 6 视频识别训练测试



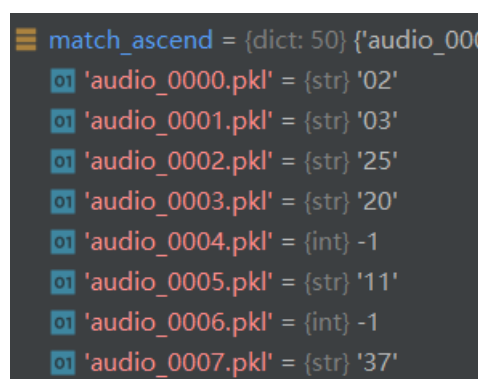
```
len(): 10
match_comb
{ 'audio_0.pkl': 'video_0', 'audio_1.pkl': 'video_1', ... }
special variables
function variables
'audio_0.pkl': 'video_0'
'audio_1.pkl': 'video_1'
'audio_2.pkl': 'video_2'
'audio_3.pkl': 'video_3'
'audio_4.pkl': 'video_4'
len(): 5
```

图 7 音频视频全匹配测试

在从训练集抽取的小型测试集上，我们的算法较好地完成了任务。

## 5 任务三

同任务二的测试，我们从训练集中选取了几对音频-视频以及没有对应的音频和视频，结果如下：



```
match_ascend = {dict: 50} {'audio_0000.pkl': {str: '02'}
01 'audio_0000.pkl' = {str: '02'}
01 'audio_0001.pkl' = {str: '03'}
01 'audio_0002.pkl' = {str: '25'}
01 'audio_0003.pkl' = {str: '20'}
01 'audio_0004.pkl' = {int: -1}
01 'audio_0005.pkl' = {str: '11'}
01 'audio_0006.pkl' = {int: -1}
01 'audio_0007.pkl' = {str: '37'}
```

图 8 音频视频不完全匹配测试

等号左边为音频文件名称，右边为视频编号，“-1”代表没有匹配项。

## 6 结果分析与存在问题

综合来看，任务一较为简单，但是由于时间较为紧张，音频识别训练结果正确率超过 90%后便没有继续调整 CNN；视频识别正确率达到了 100%。

任务二与任务三只能通过从训练集抽取数据进行小规模测试，结果没有问题，但是由于任务二的匹配算法选用的是遍历的思路，导致复杂度为 $O(n^2)$ ，运行时间很长；任务三选用的是贪心算法，复杂度约为 $O(n)$ ，运行较快一些。

在算法原理方面，在任务二与任务三的音频视频匹配中，我们最终选择了容易提取的“最剧烈碰撞”作为特征。通过对 video 的观察，我们发现物体往往发生多次碰撞，但是由于物体形状不规则，可能同时与多条边发生碰撞；以及物体速度较快，可能短时间内碰撞多次。多次碰撞在音频上容易提取，但是视频提取难度较大，因此我们最终选择只提取“最剧烈”的一次。此外，物体可能只发生了翻转而没有碰上边，这样音频的强度特征会很大但视频的强度特征会很小，这会导致匹配失败。

## 7 文件清单与使用说明

压缩包中含有 1 份实验报告以及份.py 文件。代码调用的库在 environment.txt 中列出。

all\_net\_def.py 定义了所有的 CNN；

t1\_audio\_process.py 是对四通道音频信号预处理文件

t1\_simple\_CNN\_3.py 音频神经网络的训练

t1\_simple\_CNN\_3\_test.py 对输入的音频信号特征进行前向计算，输出分类结果

t2\_audio\_amp\_feat.py 对任务 2 提取音频最大声音所在边以及最大音频功率特征

t2\_audio\_CNN\_3\_test.py 对任务 2 中的音频进行分类

t2\_audio\_process.py 对任务 2 中音频文件进行预处理，提取 MFCC 特征

t2\_im\_CNN\_1.py 图像分类神经网络的训练

t2\_im\_CNN\_1\_test.py 对任务2中图像进行分类

t2\_im\_edge\_feat.py 任务3提取视频中最大速度后的碰撞边以及运动中最大速度的特征，并且给予置信系数

t2\_set\_within\_class\_match.py 任务2对音频视频通过遍历进行匹配

t2\_top.py 任务2的顶层文件

t3\_audio\_amp\_feat.py 提取任务3中音频最大声音所在边以及最大音频功率特征

t3\_audio\_CNN\_3\_test.py 任务3中的音频进行分类

t3\_im\_CNN\_1\_test.py 对任务3中图像进行分类

t3\_im\_edge\_feat.py 任务3提取视频中最大速度后的碰撞边以及运动中最大速度的特征，并且给予置信系数

t3\_set\_within\_class\_match.py 任务3对音频视频通过遍历进行匹配

t3\_top.py 任务3的顶层文件

t1\_cnn.pth 训练好的音频分类神经网络参数

t2\_im\_cnn.pth 训练好的图像分类神经网络参数

**请务必在主调脚本中加入**

**from all\_net\_def import \***

**否则 pytorch 会找不到网络**

## 8 小组分工

小组共三人：李奥、姜安柏、熊晓桐  
三人均参与算法设计，李奥负责音频特征提取，熊晓桐负责视频特征提取，姜安柏负责2个神经网络的搭建、匹配算法的设计、所有接口的整合及代码运行测试。