音视频场景识别

521030910381-姚博

1. 引入

多模态表征学习（Multimodal Representation Learning）是指在深度学习框架下，通过整合多种不同类型（模态）的数据来学习数据的共同表示或特征表示的过程。这些模态可以包括文本、图像、音频、视频等不同形式的数据。传统的机器学习方法通常是单模态的，即只使用一种类型的数据来进行建模和预测。然而，在现实世界中，很多任务涉及到多种类型的数据，例如图像描述、视频分类、跨语言文本匹配等。在这些任务中，单独利用其中一种类型的数据可能无法充分挖掘数据的潜在信息，而多模态表征学习则可以解决这个问题。在这次任务重中，我们的数据集中包含了不同场景下的音频与视频数据，我们要经过这样一个流程来进行处理：1 特征提取，2 建立模型训练网络，考虑到场景有音频与视觉两个维度的信息，就如同时间和空间的关系，建立一个良好的特征融合方法有助于提高模型的表征能力，使其分类效果更加出色。
目前在通常的数据上，存在一些与模型无关的特征融合的方法，[h]

2. 融合方法简介

由于数据集过大，数据预处理已经实现，我们不在讨论
1. 早期特征融合：
即将预编码好的音频特征和视频特征直接拼接，传入分类器中。
2. 中期特征融合：
即将预编码好的音频特征和视频特征传入子编码网络，再将各自中间过程输出拼接，最后传入分类器整体处理。
3. 晚期特征融合：
这属于整个任务的后期阶段，在得到最终输出前进行融合，按照不同的策略进行权重调整投票并最终得到结果。

2.1. baseline 与中期特征融合

forward 函数如下：

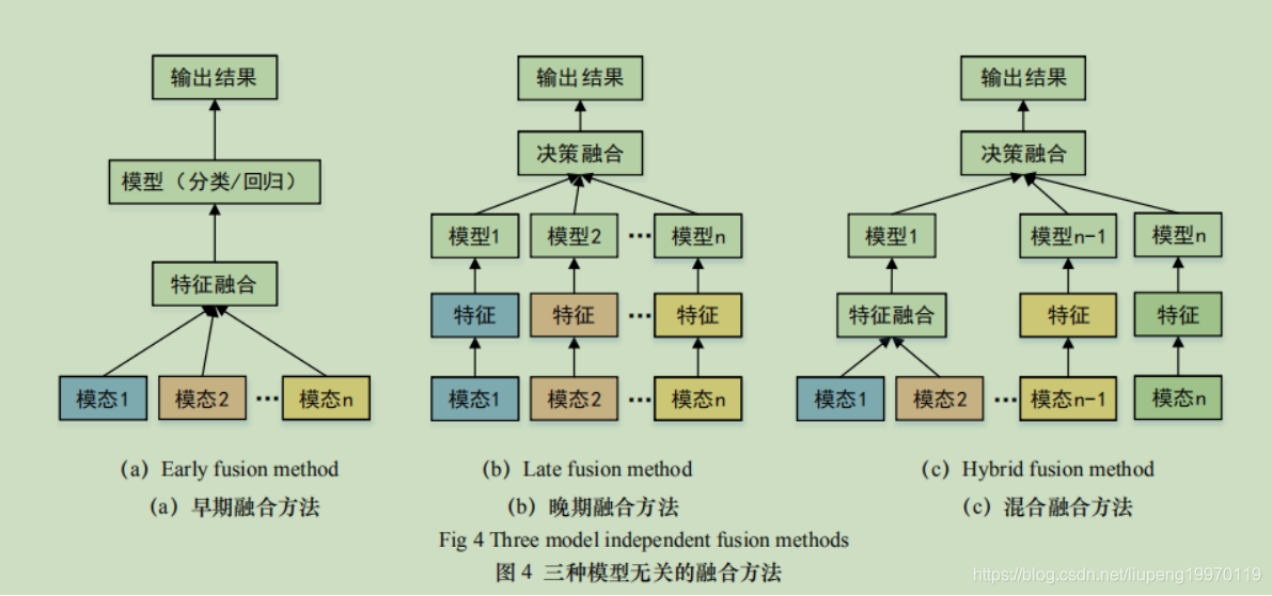
```
    def forward(self, audio_feat, video_feat):
        # audio_feat: [batch_size, time_steps, feat_dim]
        # video_feat: [batch_size, time_steps, feat_dim]
        audio_emb = audio_feat.mean(1)
        audio_emb = self.audio_embed(audio_emb)

        video_emb = video_feat.mean(1)
        video_emb = self.video_embed(video_emb)

        embed = torch.cat((audio_emb, video_emb), 1)
        output = self.outputlayer(embed)
        return output
```

具体过程则是对于输入的音频、视频三维张量，首先先降维到二维（通过在第一维度取平均），然后通过不同的编码层得到长度为 128 的张量，并将他们的第一维拼接，这样在中期得到融合特征，输入分类层，得到各类别概率张量。结果如下：
最终 accuracy: 0.813 overall log loss: 0.527

2.2. 早期和晚期特征融合

在早期特征融合，我们就是把 audio_emb_dim 与 video_emb_dim 暴力地加起来然后总体在神经网络中训练：

```
    self.early_output = nn.Sequential(
        nn.Linear(audio_emb_dim + video_emb_dim, 1024),
        nn.BatchNorm1d(1024),
        nn.ReLU(),
        nn.Dropout(p=0.2),
        nn.Linear(1024, 512),
        nn.BatchNorm1d(512),
        nn.ReLU(),
        nn.Dropout(p=0.2),
        nn.Linear(512, 256),
        nn.BatchNorm1d(256),
        nn.ReLU(),
        nn.Dropout(p=0.2),
```



(a) Early fusion method (b) Late fusion method (c) Hybrid fusion method

(a) 早期融合方法 (b) 晚期融合方法 (c) 混合融合方法

Fig 4 Three model independent fusion methods

图 4 三种模型无关的融合方法

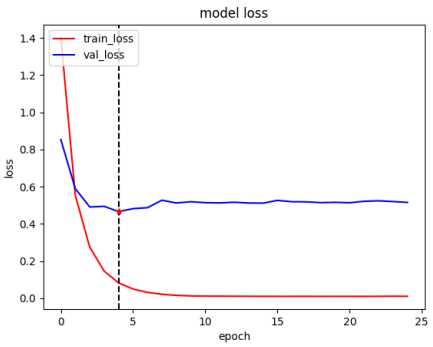


图 1: *baseline*

	precision	recall	f1-score	support
airport	0.78	0.74	0.76	281
bus	0.91	0.91	0.91	327
metro	0.87	0.83	0.85	360
metro_station	0.88	0.89	0.88	386
park	0.96	0.87	0.91	386
public_square	0.72	0.70	0.71	387
shopping_mall	0.77	0.71	0.74	387
street_pedestrian	0.70	0.72	0.71	421
street_traffic	0.85	0.95	0.90	402
tram	0.71	0.83	0.77	308
accuracy			0.81	3645
macro avg	0.82	0.81	0.81	3645
weighted avg	0.82	0.81	0.81	3645

accuracy: 0.813
overall log loss: 0.527

图 2: *baseline*

```
        nn.Linear(256, 128),
        nn.Linear(128, self.num_classes)
        )
```

accuracy: 0.807 overall log loss: 0.579

在晚期特征融合中，先单独按照自己的特征分类，最后平均或者是按权重投票：

```
    def forward(self, audio_feat, video_feat):
        #audio_feat: [batch_size, time_steps, feat_dim]
        #video_feat: [batch_size, time_steps, feat_dim]
        audio_emb = audio_feat.mean(1)
        audio_emb = self.audio_embed(audio_emb)

        video_emb = video_feat.mean(1)
        video_emb = self.video_embed(video_emb)

        audio_out, video_out = self.audio_output(
      audio_emb), self.video_output(video_emb)
        output = (audio_out + video_out) / 2
        return output
```

当然，为两个维度各自构造了一个简单的线性分类器。

accuracy: 0.828 overall log loss: 0.495

2.3. 把中期融合与晚期融合拼接一下

为了得到一个比较好的模型，我想在最终分类时，按照一定比例使用中期融合，同时考虑后期融合的信息（前期融合相当于抹杀了不同感知信息的差别，不太好）。最终效果：

accuracy: 0.822 overall log loss: 0.498

3. 结果分析

首先对于早期的特征融合，应该比较适用于相近的感官识别特征群，这样有更多共同性，对相同的数据划分更有优势，但是对不太相近的多模态任务则抹杀了不同维度的差异性。晚期特征融合则可能对更加独立的特征表示有优势，我们的数据集感觉相对而言便是均衡的既包括大量同效信息，也有独立的部分，中期特征融合则是两种方法的折合，具有更宽广的泛用性。

4. 实验总结

通过本次实验，我对多模态任务有了更深刻的认识，无论从原理上还是实践的结果上来看多模态的引入都极大地优化了场景分类任务下模型的性能。而不同的特征融合方式也对模型最终的性能有着或多或少的影响，非常感谢老师与学长的帮助!