第一次书面报告

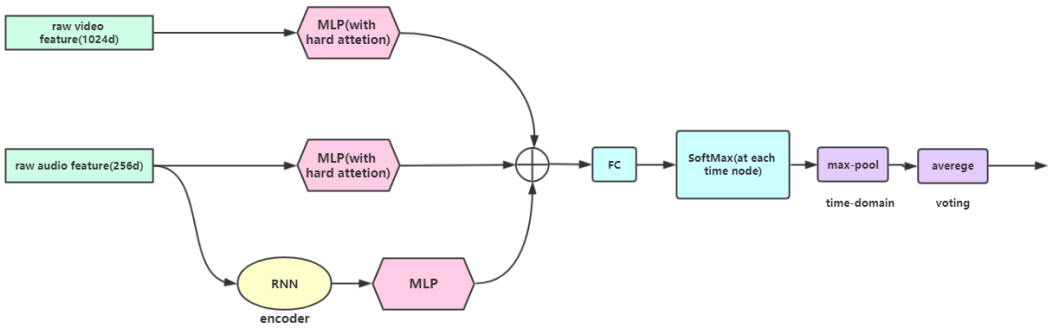


Figure 1

0. 小组成员与小组分工

小组成员：

无 52 肖善誉 2015011009

无 52 薛有泽 2015010997

小组分工：

肖善誉：主要模型设计与训练，报告撰写

薛有泽：模型设计，模型调参，额外数据提取

1. 提交文件清单

文件名	文件说明
models.py	模型定义文件
train.py	模型训练文件
evaluate.py	模型测试文件
1. 整体架构.pdf 2. MLP with hard-attention.pdf	网络结构图片
project_code	所有工程文件…

2. 原理与网络结构

2.1 总体结构

见 **Figure 1**。

（1）视频特征与音频特征分别通过基于 hard-attention 的 MLP（具体结构 **Figure2** 与之后的说明），得到相应的时域混合后的特征（time-domain mixed feature）。

（2）考虑到音频特征具有时间上的连续性（不同于视频特征是每个 1 秒采样得到的“离散”特征），我们同时将一个 bi-GRU 连接在输入音频特征后，用于将音频特征编码。在 GRU 输出后连接上两层 MLP，该 MLP 在时域上共享参数。

（3）之后将三种得到的特征相加，得到大小为 batch_size * seq * hidden_size 大小的特征。因为整个序列是否匹配具有极强的时间局域相关性，故将该特征在时域上每个位置利用 SoftMax 求得是否匹配的概

率值。

通过 MaxPool 选出固定时间段内最大的概率值（chunk-maxpool），之后把得到的多个概率值求平均（voting）。这种做法在可以减小过拟合的同时，voting 的方式也有类似 ensemble 的效果。

2.2 MLP with hard-attention

见 **Figure 2**

图中以 1024 维的视频特征为例。

因为整个序列是否匹配具有极强的**时间局域相关性**，在同一时间点附近的视频、音频特征相关性很强，因此我们在网络中引入了 **hard-attention** 机制，设置一个在时域上的滑动窗口，来混合不同时间点上的特征。

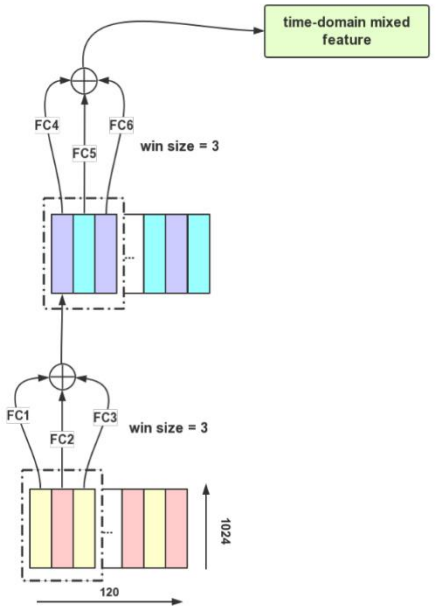


Figure 2

在每个时间点上为一个 1024 维的视频特征。设置滑动窗口大小为 3，并设置了两层的局部结构，因此最后**得到的特征是该时间点附近 5 个时间点的特征的组合**。

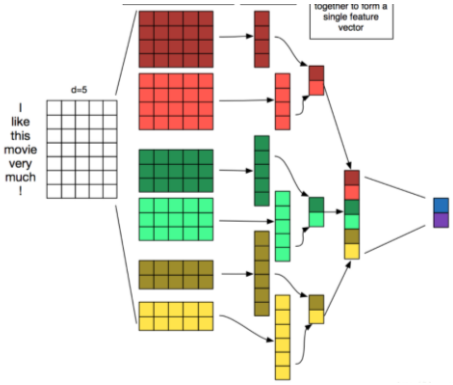
对于每个窗口内的每一个视频特征，它们连接不同的全连接层（图中第一层的 FC1，FC2，FC3 与第二层的 FC4，FC5，FC6）。之后将全连接层的输出相加，得到在时域上混合的特征。滑动窗口在不同位置处，全连接层的参数共享。

通过这种结构，我们就可以达到在时间局部上计算相关性。因此，利用得到的时域混合特征，在每个时间节点上计算匹配概率（利用 Softmax），得到一个大小为（$Batch120 \times 2$）的匹配结果（概率）。

MLP with hard-attention 结构可以有一种等价的 CNN 描述方式。我们可以把经过 MLP 后的视频、音频特征，经过 RNN 后的音频特征视为一张图片的三个 channel。将这三种特征的长度都输出为一个值时，如 128，我们可以把整个视频音频视为“一张图片”，大小为$3 \times 120 \times 128$。

这之后，将“图片”通过两层 keinel_size 为3×128的宽卷积层，皆可以得到与 MLP with hard-attention 相同的结果。

这也类似于 NLP 中 CNN 的应用，将词 embed 为多维词向量后，将整个句子视为一个“图”，然后利用 CNN 进行处理。图示如下：



2.3 Max-pool 与 average

对于 120 个时间点上的匹配结果，以 kernel_size = 5，选出每个时间区域内匹配结果的最大值（**chunk-maxpool**），得到 24 个匹配结果。然后对这 24 个值求平均，得到最终的匹配概率。

采取这种方式的主要原因有 2 个：

1. 减少过拟合：即使整个视频与音频相互匹配，在各个时间点上也不一定完全匹配。因此只要求每个 chunk 内某个（些）位置比较匹配即可。否则很容易造成模型过拟合。

2. 将 24 个值求平均，是一种 voting

机制。如果视频与音频相互匹配，则在整个时间序列上相关性应该都比较强，因此平均后的结果应该比较接近 1。Voting 某种程度上也可以减少过拟合。这是综合全局结果的一种方式。

2.4 实施细节

最终测试时使用的模型，**综合了两个单模型的结果（ensemble）**，即：使用两个网络结构相同的模型，对他们的结果取平均值。使用这一方法，top5 准确率上升了 3.3%，达到了 96.7%.

使用的 RNN 为双向 GRU，层数为 3。

激活函数采用 scaled hyperbolic tangent function :$f(x) = 1.7159 \cdot tanh\left(\frac{2}{3}x\right)$, 以加快训练速度。

最终的损失函数采用带有 focal loss 的交叉熵损失函数，来对难以区分的样本给予更大的权重。另一方面，这也相当于一种 soft 的 margin loss，通过降低结果比较接近 0 或 1 的样本的损失权重，达到 margin loss 的效果。

本实验中使用的损失函数定义如下：

$$p_t = \begin{cases} p, y = 1 \\ 1 - p, y = 0 \end{cases}$$

$$L = -(1 - p_t)^2 \cdot log(p_t)$$

产生负样本时，利用源代码中的相同方法，产生两倍于正样本数的负样本，以增加样本数量。

2.5 其他额外尝试

匹配或不匹配可以对应于一个二分类问题，因此实验过程中尝试了以神经网络提取特征，利用经典的分类器来进行分类。

我们尝试了使用 boosted tree（包括 Adaboost, Random Forest, GBT 等）作为最终的分类器，但实验发现，top5 的结果最高只能到达 83.3%，不如直接使用全连接层与 Softmax 效果好，因此放弃了这种方法。

产生负样本时，在每个 batch 内打乱顺序的基础上，尝试利用在每个样本时间上打乱产生负样本。发现这样会使得测试准确率下降，原因猜测是这种负样本无法准确代表样本集的分布。

3.实验结果与性能分析

最终 **top5** 准确率为 96.7%。

top1~top5 准确率如下：

Top k	准确率(%)
1	60.0
2	70.0
3	83.3
4	93.3
5	96.7

可以看到，基于局部相关的 attetion 机制的模型能够比较有效地对将视频音频进行匹配。

性能分析：

由于数据集样本数量比较少，因此我们倾向于使用容量比较小的模型，来防止过拟合，MLP with hard attention 的层数仅为 2，不算太深的网络，因此运行速度不算太慢。

匹配 30*30 = 900 的视频音频对，耗时 0.47448s，平均每对匹配花费 0.5272ms。

测试后发现，主要的性能瓶颈在于 RNN(GRU，层数为 3， 双向)。RNN 迭代 120 个时间周期，运行时间相当于层数为 120*6 的全连接层.因此主要耗时在 RNN 上。

4.问题与不足

1. 实验中发现，每经过一个 epoch 的训练后，测试结果都可能有比较大的变化。例如，上一个 epoch 训练后的测试结果可能达到 90%以上，而下一个 epoch 训练后结果仅有 76.7%。导致这种现象的原因，其一是训练数据集与测试数据集比较小，可能导致比较大的测试方差。另一方面，说明模型部分程度上稳定性不够好。这个问题也许可以通过手动提取更多数据来减轻。

2. 模型对于视频音频的整体匹配的考虑不够多。我们的模型主要利用的是时间上局部匹配，再通过 max-pool 和 average 的方法来达到全局考量的目的。这样可能导致对整体的考量不够多。

3. 由于 max-pool 有使得输出结果趋向于 1 的倾向，因此发现模型最后 30 * 30 矩阵中的值都趋向于高值（如 0.7， 0.8），即倾向于输出不匹配。同时匹配与不匹配的结果相差比较小。因此，我们的模型在已知很对多视频音频中有一对匹配，其余不匹配的前提下，得到不同视频音频匹配结果，并比较他们的值时，才能有比较好的表现（本实验中的场景）；但如果单独拿出一对视频音频，并决定他们是否匹配，我们的模型就会表现很差。