视听信息导论大作业报告

邓博文	魏旭	张耀予
2015010999	2015011105	2015010998

2017 年 12 月 20 日

目录

1 思路

本次大作业给定的任务是使用机器学习的方法，利用给定的 1300 组数据，设计一个神经网络，使其能够评估给定的无声视频与音频的匹配度，进而找回每个无声视频所对应的原始音频文件。一开始，我们小组成员基于自己对问题的理解，提出了两种基本的网络结构设想。

CNN 将输入的视频特征和音频特征都看成通道数为 1 的图片，并让它们分别通过经典的 CNN 结构，对二者进行进一步的特征提取，最后综合两者特征即可。

RNN 因为输入的 vfeat 和 afeat 是对视频逐帧采样得到的，所以自然具有时间序列的特性，即同一个视频采样得到的视频音频特征在时间轴上前后是有联系的，而 RNN 可以充分利用这种关联，比较方便。

CNN 具有很强的特征提取能力，RNN 则可以充分利用序列在时间上分布的关联，二者各有所长，我们希望网络可以同时具有二者的长处，但是这种结构的难点在于网络具体结构的确定，为此，我们决定先实现 CNN 和 RNN 的架构，再尝试二者可能的搭配形式，进而寻求最优的网络结构。

2 结构设计

2.1 探索

我们根据我们的思路进行了很多探索，探索的模型和效果参见附录D。

2.1.1 双路 CNN

我们从一般人比较两个序列的方式上去设计我们的网络结构，即对于给定的序列，应该独立的进行处理，然后再放到一起去判断。这类似于 Siamese 网络结构, 但是由于音频、视频大小、性质完全不同，我们无法用结构相同、共享权重的网络来统一处理音频、视频，但可以尽量保证结构类似。首先我们采用最常见的 CNN 模块组合（$Conv \to BatchNorm \to ReLU \to MaxPool$），让视频音频独立过两个多层 CNN 后输出得到同样长度的特征向量，之后并联传递给一个全连接网络，结果不错，大概达到 75%2。

2.1.2 双路 RNN

基于同样想法，我们设计了这样一个结构：让音频，视频分别经过两个 LSTM 网络（分别称 $alstm$、$vlstm$），然后，将输出序列在特征维度上拼接起来，让拼接后的序列再经过一个 LSTM 网络 (称 $olstm$)，最后一个时间片输出通过一个全连接层得到输出。收敛后测试准确率大约在 75%3。

2.1.3 残差 RNN

既然已经获得了比较好的效果也没有出现过拟合，我们尝试将网络变得更深一些，毕竟我们这个 RNN 结构2.1.2相比上文 CNN 结构2.1.1来说太浅了。我

们首先尝试给 olstm 再加一层，但是结果反而变得很差, 仅有 50%。按道理，更深的网络表达能力更强，而不应该更差，于是我们借用残差网络的想法 [4]，给三个 LSTM 之后各增加一个 $Dropout \rightarrow LSTM$，并用残差网络的方法旁路连接输出。结果可以达到 80%4。

2.1.4 Attention

Attention 结构已经在机器翻译领域取得很大成功 [1]，在这里同样也可以利用它来让 RNN 更好地处理每一帧之间的信息，但是我们并没能成功找到一种结构有接近上文结构的准确率。究其原因很有可能在于结构中产生 Attention 的结构频繁使用，难以训练。[1]

2.2 综合

我们还注意到 RNN 结构输出具有这样一个问题，即 RNN 只能将最后时间点的输出给予之后的全连接层2.1.2，如果将输出序列全部连出来，全连接层会过大。我们想到可以在这里应用 CNN 结构2.1.1，它正是输入一个序列而输出一个不大的特征向量。如此，我们大概确定了我们的结构[2]，即 $RNN \rightarrow CNN \rightarrow FullyConnected$。并且运用残差网络的想法，使 RNN、RNN 都加深。最终结果如图1。

3 调参

在之前结构设计部分的基础之上，经过多次测试，我们选择了上述结构1作为我们最终的网络结构，为了进一步提高网络的性能，我们开始对网络的参数进行调整. 这一过程大概分为如下几部分。

3.1 参数初始化

参数初始化对于一个神经网络来说是极其重要的一步。不恰当的参数初始化可能使得各层的输入的方差过大，当网络很深时, 这很可能导致反向传播时过早地出现梯度消失等问题，好的初始化应该可以打破网络的对称性并避免梯度消失。经过上网查阅相关资料，我们发现，使用 $Xavier$ 初始化 [2] 可以使得每一层的方差尽量相等，从而使得信号经过多层神经元后依然保持在合理的范围，进而避免了前向传播爆炸和反向传播梯度消失等问题。

3.2 优化器

不同的优化器之间难以直接比较，很难说哪一个的效果一定最好。结合我们实际的网络结构，经过尝试后，我们发现，使用 SGD、$RMSProp$ 都存在收敛比较慢的问题，为了减少训练所需 epoch，我们最后选择了 $Adam$。

[1]我们之后也尝试了 Attention 后连接 CNN 的结构，得到了比单独使用 Attention 相比更好的结果，但还有很大问题。我们推断 Attention 似乎不能发挥效果，就没有继续尝试。

[2]RNN 输出全部接出的情况用虚线表示（附录D同理）。

3.3 误差函数

我们在调试中发现，助教给定的 $ContrastiveLoss$ 似乎并不适合于我们的模型。事实上，我们的模型后部将两者整体进行计算的过程不满足论文 [3] 的假设，即我们网络的输出不是简单的欧氏距离，所以也就不能运用这种误差函数。考虑到神经网络可以近似几乎一切连续函数，那么我们结构后部已经足够拟合出一个度量函数了。据此，我们不需要给予网络太多的限定，只要最简单的 $BinaryCrossEntropy$ 就可以满足我们需要。

3.4 学习速率

训练参数中最为重要的就是学习率。学习率是否恰当可以说决定了一个网络的最终表现甚至能否收敛。但是学习率大小的选择并没有一定之规，更多的是尝试。为了能够找到恰当的学习率取值，我们又单独写了一个文件，选取了 3 个量级 $10^{-1}, 10^{-2}, 10^{-3}$，每个量级选取了两个值，来进行测试，最终发现，设置学习率为 0.005 时，可以得到最高的准确率。此外，为了使得网络在经过较多 epoch 后，仍能以比较适宜的学习率训练，我们使用了 $ReduceLROnPlateau$ 函数，当经过若干周期网络表现没有提升时，动态地减少学习率。

3.5 结构细调

注意到网络结构本身也有很多需要调参的地方，我们在模型收敛速度、过拟合、模型大小之间努力寻找折中，最终结构1深度与宽度的配合就是这么选出来的。但时间仓促，我们没能更精细的调参，很多都采用默认值（$Dropout$ 中 $p = 0.5$、$LeakyReLU$ 中 $negative_slope = 1 \times 10^{-2}$ 等）或直接去掉（$BatchNorm$ 的 $affine$ 等），但已经能达到足够好的效果。

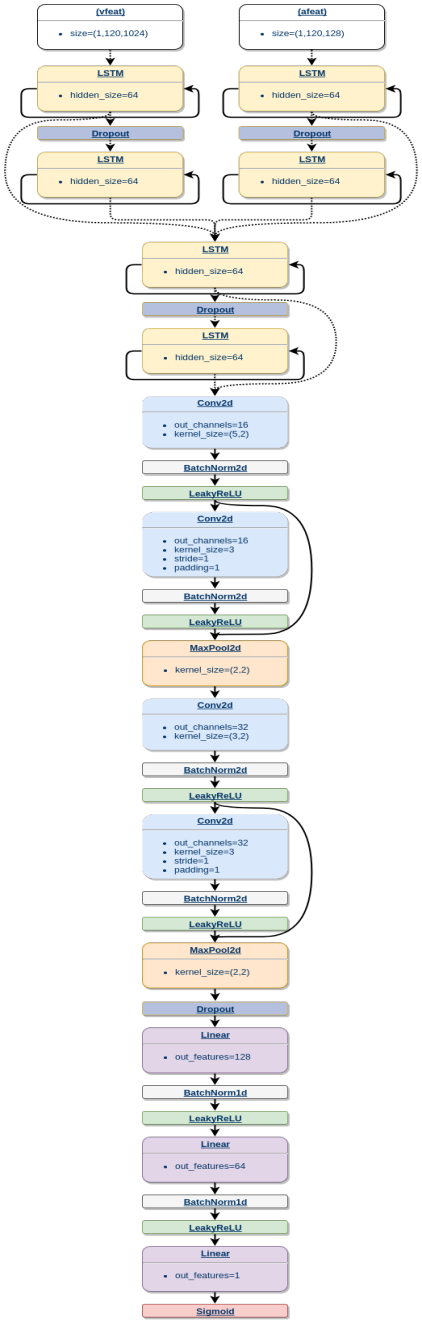
4 结果及分析

我们所选择的模型最终结果如图1，注意到我们的测试准确率收敛时都达到了 80% 以上，如果考虑最大值，普遍都可以 86.7% 乃至 93.3%，这与附录D中各个模型相比，都有较大的进步。

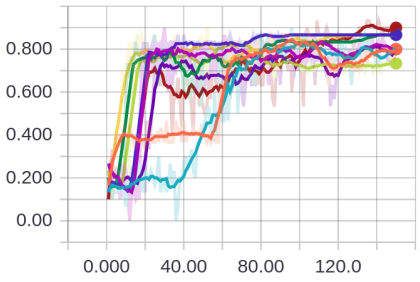
最终我们提交了准确率最高的模型，准确率达到 *100%*[3]。这看似不是必然（考虑到网络初始参数随机，训练样本随机等事实）但我们后来敏锐地发现这么一个问题：我们的网络最终会十分稳定，而学习率因为单调减地动态调整*3.4*已下降得很低。我们认为有这样一个可能，即学习速率帮助了模型的稳定，但很容易导致陷入局部最优。[4] 并且我们还注意到，这个局部最优（*86.7%* 上下）似乎是很常见的，既然大量的随机性仍然不能让模型离开这里，我们可以认为它附近有更好的最优值，只是学习率已经太小达不到而已。对于已训练好的网络（准确率 *90%* 以上），我们强制调整学习率至 *0.001*，继续学习，便达到了更好的解。

[3] 我们绝对没有把测试集当作训练集，这的确是我们最好的结果。

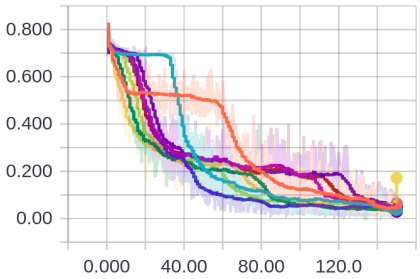
[4] 图1中训练集明显的阶梯形状应该就与学习率有关。



(a) 网络结构



(b) 测试集正确率



(c) 训练集误差

图 1: 最终结果

5 总结

这次大作业出题新颖，涉及面广，尤其是与当下最火的深度学习紧密相关，所以我们也抱有很大兴趣去尝试各种各样的想法。同时，因为大作业内容与课程内容结合紧密，使得我们可以将课上老师讲授的内容应用到实际问题的解决中去，这既让我们能够对课上的知识深入理解，又使我们在此基础上更进一步，打破了我们对深度学习就是”炼丹术”、”黑盒子”的偏见，让我们体会到了自己设计网络的快乐与充实感，总的来说，虽然耗时良多但收获很大。

首先，我们组都是初次接触深度学习，以前从来没有自己尝试过设计结构，也没有很多调参经验，这次能有机会一起相互学习进步非常难得，我们也积累了一些对以后来说很有用的经验。技术方面，我们采用了 Pytorch 框架，在 Ubuntu 平台上运行程序，对自己的代码能力、调试能力有了很大锻炼。研究方面，为了能更有思路，我们了解了很多相关研究成果，对该领域形成了初步的看法，对以后的科研和应用打下坚实的基础。

时间有限，我们也许可以做的更好！谢谢老师、助教以及微信群里积极讨论的同学对我们到作业的完成提供的帮助！

参考文献

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *international conference on learning representations* (2015).

[2] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: (2010), pp. 249–256.

[3] Raia Hadsell, Sumit Chopra, and Yann Lecun. “Dimensionality Reduction by Learning an Invariant Mapping”. In: 2 (2006), pp. 1735–1742.

[4] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *computer vision and pattern recognition* (2016), pp. 770–778.

A 小组分工

任务	负责
网络构思与搭建	魏旭、张耀予、邓博文
网络结构优化与调参	魏旭、张耀予、邓博文
作图与报告撰写	魏旭、张耀予、邓博文

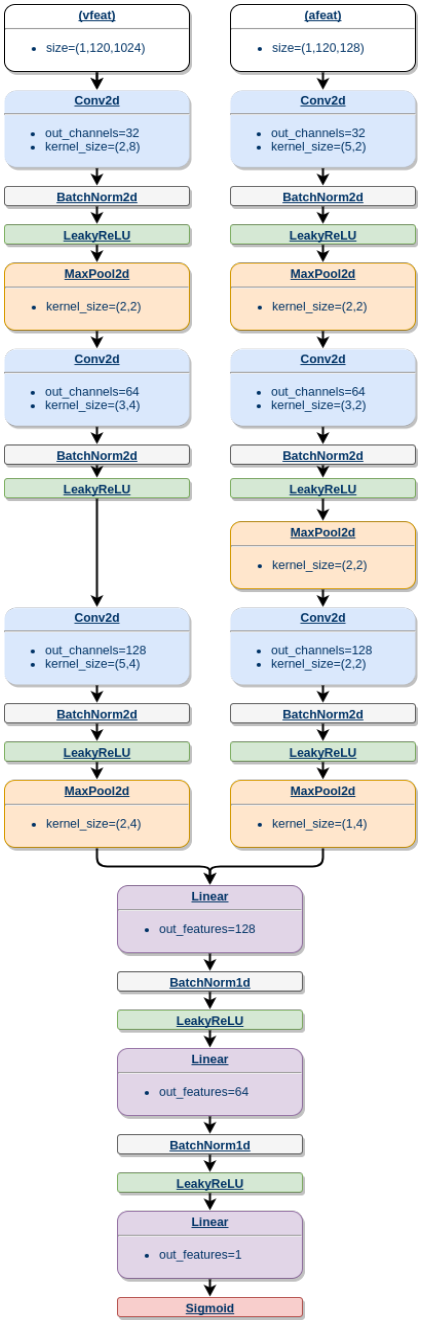
B 依赖清单

名称	版本	用途
python	3.6.3	Python 解释器和标准库
numpy	1.13.3	矩阵运算、存取
pytorch	0.3.0	网络训练
tensorboardX	0.8	训练可视化
tensorflow	1.3	tensorboard 界面

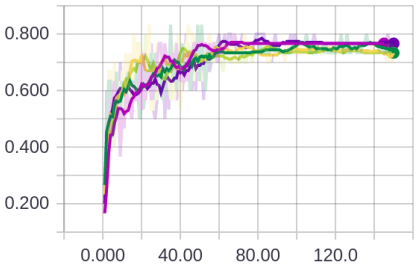
C 文件清单

```
./
├── checkpoints/
│   └── ResidualRNNConv.pth ..... 训练好的模型
├── configs/
│   └── config.json ..... 参数与配置
├── filelists/ ..... 训练集、测试集划分
├── tools/
│   ├── dataset.py ..... 数据集处理
│   └── utils.py ..... CUDA 环境配置等（没有使用 CPU）
├── evaluate.py ..... 模型测试
├── models.py ..... 模型定义
└── train.py ..... 模型训练
```

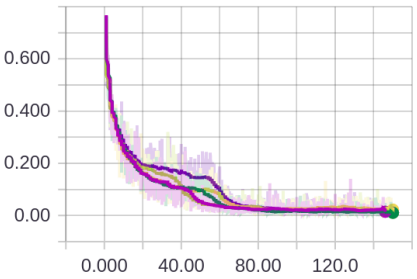
D 尝试模型细节



(a) 网络结构

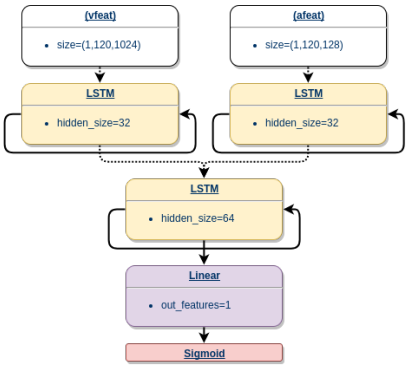


(b) 测试集正确率

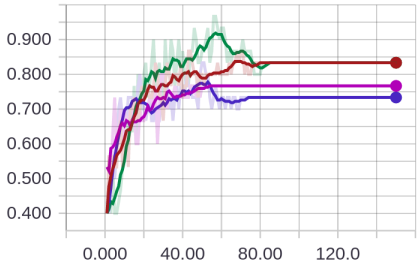


(c) 训练集误差

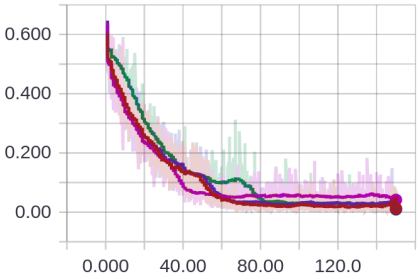
图 2: 双路 CNN



(a) 网络结构

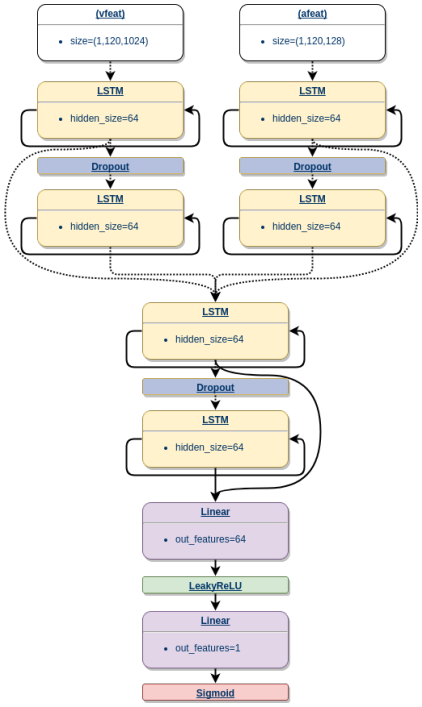


(b) 测试集正确率

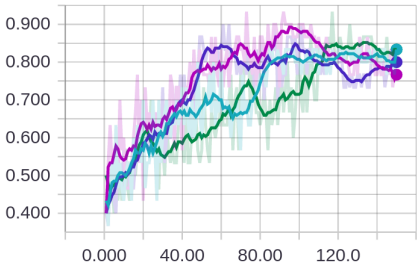


(c) 训练集误差

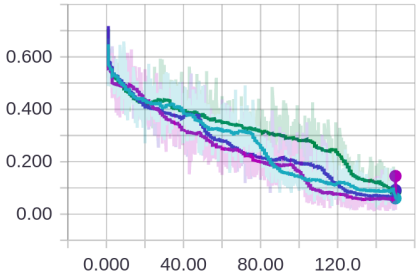
图 3: 双路 RNN



(a) 网络结构



(b) 测试集正确率



(c) 训练集误差

图 4: 残差 RNN