声音事件检测 (Sound Event Detection)

519030910360 郑文卓

1. 简介

本项目为声音事件检测 (SED)，我的主要工作如下:

1. 调研声音事件检测的背景，了解弱监督情况下进行时间轴预测的难点以及 baseline 的设计原理；
2. 理解代码逻辑，并按照 CRNN 框架实现和改进了模型；
3. 修改模型深度、参数、超参数等以优化模型性能；
4. 调研和学习 SED 中的数据增强方法并应用。

项目代码见`https://github.com/darkcorvushhh/Sound-Event-Detection`。

2. 背景介绍

2.1. 定义

一般而言，声音事件检测的目的是识别音频信号中发出声音的事件是什么，以及该事件发生的时间 [1]。如图1所示，其任务可以分为两部分: 为事件分配标签 (tagging) 和为事件定位起止时间 (localization)。

图 1: 声音事件检测

在具体实现时，通常会将时间轴划分为连续的时间单元 (如图2)，SED 旨在找到声学场景中的短时活动，以时间单元为颗粒度判断其类别以及时间。从这个意义上说，对于每一个时间划分而言，SED 是一种多分类多标签的任务。

目前主流的方案都是利用深度模型进行声学建模，而训练模型的方式主要有两种: 有监督的和弱监督的。

如图3所示，有监督的训练，一般会详细地标注每个事件的时间戳和持续时间，而这需要大量且昂贵的人工时间轴标注；弱监督的训练，一般只给定了语音信号中的事件标签，而并没有与之相关的时间信息，但却需要在推理阶段预测出事件的起止时间，从而带来较大的挑战 [2]。

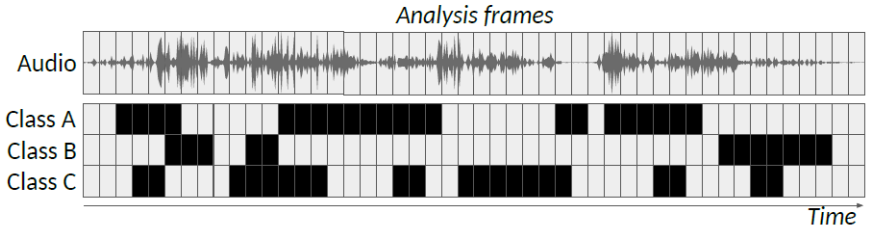

图 2: 时间轴划分

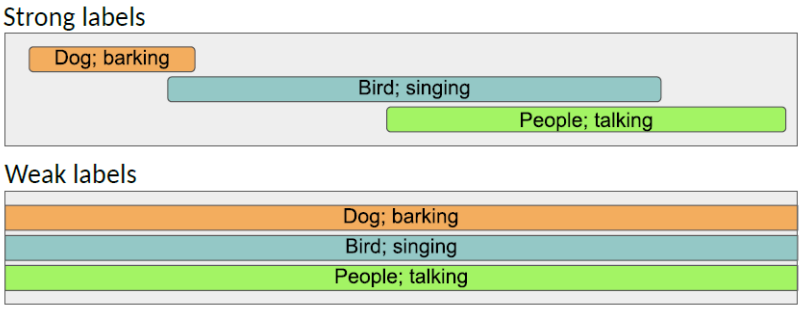

图 3: 有监督和弱监督

2.2. 难点

目前，声音事件检测中主要有以下几个难点:

1. 声音事件的声学特征非常多样，各个声音事件的长短、音调、音色各不相同;
2. 自然环境一般会有多种声音混杂在一起，同一时间有多个声音事件一起发生;
3. 声音事件的种类复杂多样，数量庞大;
4. 目前声音事件的分类并没有统一且有效的通用描述，分类的定义大多模棱两可。

而对于弱监督学习而言，由于关键的时间信息的缺失，其对事件起止时间的定位更为复杂，模型需要在缺少指引的情况下学习时间信息，这让声音事件检测更加复杂。

2.3. 流程

声音事件检测的一般流程如图4所示，主要包含两个阶段: 学习阶段和测试阶段。学习阶段会提取音频特征，根据标签训练声学模型，测试阶段则根据训练好的声学特征来对提取的特征进行分类。

2.3.1. 特征提取

在 SED 中使用最广泛的特征是对数梅尔频谱 (Log Mel Spectrogram, LMS)[3]。研究表明 [4]，人类对频率的感知并不是线性的，并且对低频信号的感知要比高频信号敏感。例如，人们可以比较容易地发现 500Hz 和 1000Hz 的区别，却很难区分 7500 和 8000Hz。基于此，梅尔标度 (the Mel Scale) 被提出，它是 Hz 的非线性变换，对于以梅尔标度为单位的信号，人们对于相同间隔的信号的感知能力几乎相同。一个常用的变换公式为:

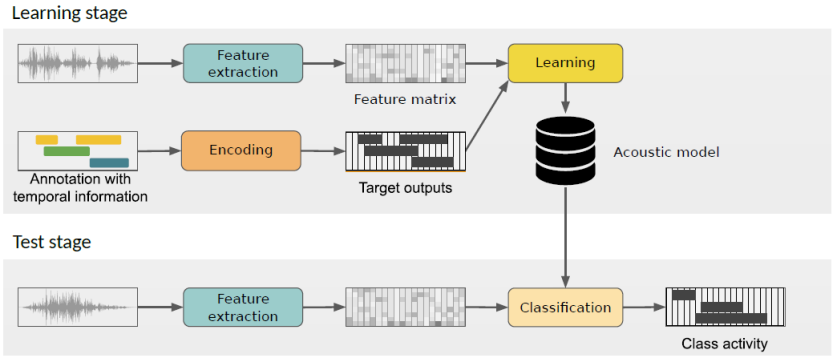

图 4: 声音事件检测流程

$$mel(f) = \frac{1000}{\log 2} \log(1 + \frac{f}{1000}) \tag{1}$$

这一过程是通过一系列的梅尔三角滤波器实现的，而梅尔频谱 (Mel Spectrogram) 就是在梅尔标度下的频谱。一般而言，人们会对得到的系数取 log，从而得到特征 LMS。

2.3.2. 模型结构

在提取了 LMS 特征后，这些特征会用来训练一个声学模型，而目前最常用的模型结构是卷积循环神经网络 (Convolution Recurrent Neural Network, CRNN)。

如图5所示，CRNN 主要包含 CNN 层和 RNN 层。CNN 层用来特征提取，其目的是学习更具有鉴别性的特征；RNN 层用来学习特征之间的时间依赖关系，其目的是得到时间上的语义信息。

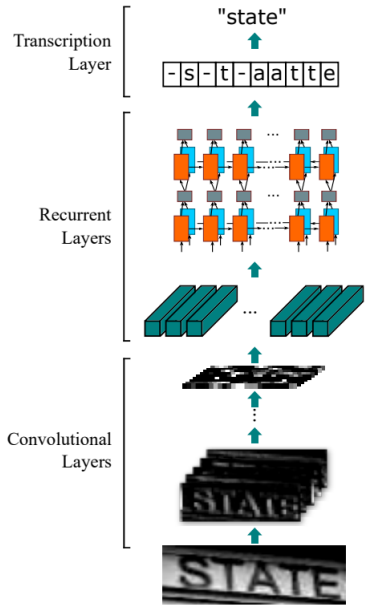

图 5: *CRNN* 模型结构

在经过 CNN 和 RNN 后，我们需要将学习到的特征转化为我们所需的输出。SED 中一般需要两种输出: 每个时间划分上的预测，即事件发生的时间，以及整段语音的事件标签，即事件发生的类别。在本次试验

中，模型采用了 FC 层和 Sigmoid 层将特征转化为了时间轴上的预测，再利用 Linear Softmax(LinSoft)[5]得到事件的标签。

LinSoft 的公式如下:

$$y(e) = \frac{\sum_t^T y_t(e)^2}{\sum_t^T y_t(e)} \tag{2}$$

它不需要学习参数，可以被视作一种自身权重平均算法。

2.3.3. 模型训练

由于弱监督学习中只提供了语音的事件标签，而没有对应的时间信息，因此训练时只能对预测的事件标签求梯度并反传从而更新模型参数，而时间轴预测仅仅用来评估指标。

本次实验采用的 loss 函数为 Binary Cross Entropy Loss(BCEloss), 定义为:

$$\mathcal{L}(y, \hat{y}) = -\hat{y}\log(y) + (1-\hat{y})\log(1-y) \tag{3}$$

BCEloss 鼓励模型的预测结果更加贴近真值，是常用的 loss 函数。

2.4. 指标

本次实验共采用三种指标:

1. **Tagging-F1 Score**: 衡量模型是否准确预测了语音中出现的事件的标签；
2. **Seg-F1 Score**: 如图6所示，以可调节的时间分割为尺度，衡量模型的时间定位能力；
3. **Event-F1 Score**: 如图7所示，衡量模型是否能准确定位事件发生的起止时间的能力。

由于本次实验更关注模型能否在弱监督情况下进行鲁棒的时间预测，因此 Event-F1 Score 作为本次主要参考的指标。

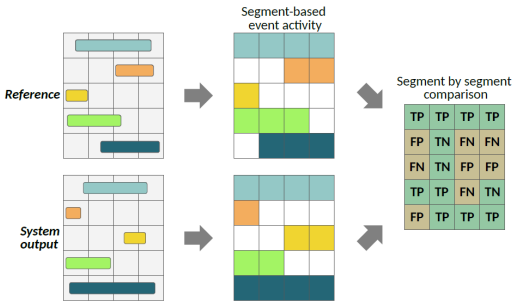

图 6: *Seg-F1 Score* 评估流程

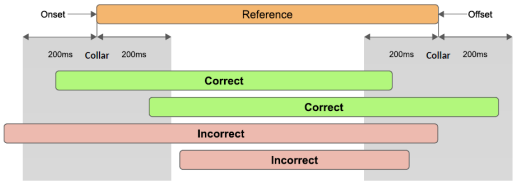

图 7: *Event-F1 Score* 评估流程

3. 模型实现和改进

3.1. baseline

本次实验，我首先根据图8搭建最基础的 CRNN 网络，但实验证明其效果并不好。

分析原因，我认为在卷积层的每一个模块的池化过程使用 2*2 的 kernel size 并不是一个较好的选择，因为该操作让特征向量在时间维度逐渐缩减，将不同时间的信息杂糅在一起，而这导致最终提取到的特征在时间上没有很好的鉴别性，尤其体现在 event 和 seg 指标上，因为这两个指标检验模型的时间定位能力。

基于此推测，我进行了改进实验，将池化的 kernel size 改变为 1*2，即在时间尺度上保持特征的一致性，二者的性能如表1所示。

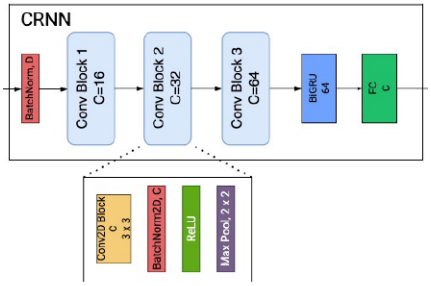

图 8: *CRNN baseline* 框架

	event F1	seg F1	tag F1
baseline	0.006	0.170	0.639
pooling[1]	0.078	0.592	0.631

表 1: *baseline* 和 *pooling* 对比

[1] pooling 表示使用 1*2 的 kernel size

从表中可以看到，相较于 baseline，改变了池化的 kernel size 之后 event 和 seg 的指标都有较大的提高，这证明我分析的原因是正确的。

3.2. 插值

查阅资料 [2] 时我发现，解决上述问题的手段除了修改 kernel size 以外，还可以在模型的最后进行上采样，从而得到与原来时间维度相同的特征。于是我在 baseline 的基础上进行插值，使用的函数为 nn.interpolate，插值模式为 linear。

但随后的实验中我发现，使用该插值模式会导致得到的结果高度随机，哪怕使用相同的参数和模型也无法得到相同的结果。随后我查阅了相关文档和文献，在 PyTorch 的官方文档中找到了原因: 该插值函数在 GPU 上可能产生不稳定的梯度，从而使结果可能具有随机性。

在经过漫长的探索后，我发现将插值模式改为默认的 nearest 时，结果能够固定，且效果与 linear 相差无几，由此，我使用插值模式 nearest 进行后续的实验，结果如表2所示。

	event F1	seg F1	tag F1
baseline	0.006	0.170	0.639
interpolate	0.114	0.616	0.681

表 2: *baseline* 和 *interpolate* 对比

可以看到相较于 baseline，插值在三个指标上都有着较大的提升，这说明插值让时间上融合的特征重新具有区分度，且从中提取出了更鲁棒、更具有鉴别性的特征。

3.3. 特征维度

baseline 中使用的特征维度是 64，而如果提高特征维度，是否能提高模型性能呢？抱着这个疑问，我提高了特征维度为 256，结果如表3所示，可以发现特征维度的提高带来了 event F1 的明显提升，而另外两个指标也相差无几，说明提高特征维度让模型有更好的检测能力。

	event F1	seg F1	tag F1
pooling-64	0.078	0.592	0.631
pooling-256	0.152	0.571	0.650
interpolate-64	0.114	0.616	0.681
interpolate-256	0.174	0.610	0.669

表 3: 特征维度变化对比

3.4. 数据增强

3.4.1. 简介

数据增强可以很好地提高数据的数量和多样性，让模型学习到更鲁棒的关系，缓解过拟合现象。在 SED 中，常见的数据增强方式有: 添加噪声、与不同冲激响应卷积、对音频数据进行组合、时间拉伸等。其中时间拉伸和音频融合的示例如图9所示。

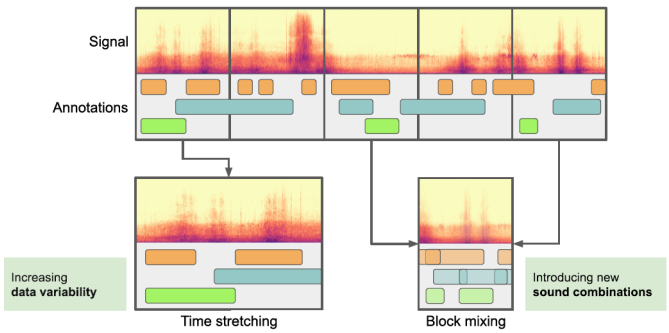

图 9: 数据增强

本次实验我尝试了以下的数据增强方式:

3.4.2. Time Shifting

Time Shifting 思路非常简单，就是将音频特征在时间上往前或往后移动随机步长，这样可以有效提高数据的多样性，移动的时间则从正态分布 $\mathcal{N}(\mu, \sigma)$ 采样得到。

3.4.3. SpecAug

SpecAug 是一种比较简单、耗时较短的数据增强方式 [6]，广泛用于端到端的语音识别任务。它直接作用于时域和梅尔频谱，利用随机生成的 time mask 和 frequency mask，在时间上或频率上置零对应区域，可以大大提高数据的多样性。

3.4.4. 实验结果

实验结果如表4所示，统一采用 learning rate 为 0.0005 和卷积 kernel size 为 11。

	event F1	seg F1	tag F1
interpolate	0.137	0.578	0.645
interpolate+freq	0.127	0.555	0.630
interpolate+time	**0.150**	0.578	0.640
interpolate+shift	0.142	0.580	0.646
interpolate+time+shift	0.145	0.580	0.640
interpolate+full	0.124	0.543	0.615

表 4: 数据增强方式对比

可以看到对于 event F1 Score 而言，表现最好的情况是只使用 time mask，而使用 freq 和 shift 后效果提升不明显，甚至有所下降，因此本次实验最终只采用了 time mask 的数据增强方式。

3.5. 参数调整

为了达到较好的性能，我还调整了多种参数，比如 GRU 层数，learning rate 的调整，卷积层的 kernel size 等，最后最优的配置为：GRU 层数 nl 为 2，学习率 lr 为 0.0006，卷积核大小 ks 为 3，结果如表5所示。

	event F1	seg F1	tag F1
baseline	0.006	0.170	0.639
interpolate+256+time	0.188	0.590	0.680

表 5: *baseline* 和最优性能对比

3.6. 其他探索

在实验过程中，我还进行了其他方向的探索。

3.6.1. LSTM

查阅资料，LSTM[7] 同样是一种 RNN 结构，一般认为它相较于 GRU 而言参数更多，结构更复杂，但效果更好。本次实验也探究了利用 LSTM 搭建网络，但最终发现其表现并不好，推测原因可能是 LSTM 结构过于复杂，在弱监督的情况下难以进行有效的训练。实验结果如表6所示。

	event F1	seg F1	tag F1
gru[1]	0.169	0.599	0.678
lstm[1]	0.157	0.580	0.666

表 6: *baseline* 和最优性能对比

[1] 参数为 lr0.0005+ks3+nl2

3.6.2. 不同融合策略

在卷积层和 RNN 层的过渡阶段，需要将特征的 channel 和 feature 维度融合，这里可以使用 PyTorch 提供的 reshape 或 flatten 函数。从原理上理解，这二者应该没有区别，或区别较小，但事实上最后的结果有明显区别，结果如表7所示。

	event F1	seg F1	tag F1
pooling1[1]	0.157	0.602	0.645
pooling2[2]	0.142	0.533	0.615

表 7: 不同融合方式对比

[1] 使用 reshape，参数统一为 lr0.0005+ks13+nl1
[2] 使用 flatten

推测是由于 PyTorch 的底层实现不同导致的差异。由于 reshape 表现更好，在选择融合方式时如无特殊说明，均采用 reshape。

3.6.3. 不同运行环境

在实验过程中，我在不同的服务器 (实验室/超算) 上使用相同的音频数据、相同的数据处理脚本、相同的训练代码进行训练，训练的结果却完全不同，表8体现了这种差异。我还与其他同学合作进行了探究，相同的代码却在四种环境中得到了四种截然不同的结果，event F1 Score 从 0.16 到 0.21 不等。

在进行反复试验后，我认为原因可能是运行环境不一致。虽然我对主要的库的版本都进行了对比，如 librosa，PyTorch，版本都是一样的，但可能仍有部分库版本不相同。

	event F1	seg F1	tag F1
baseline1[1]	0.006	0.170	0.639
baseline2[2]	0.007	0.172	0.659
interpolate1[3]	0.137	0.578	0.645
interpolate2[3]	0.145	0.569	0.645

表 8: 不同服务器对比

[1] 实验室服务器
[2] 超算平台
[3] 参数统一为 lr0.0005+ks11+nl1

3.6.4. 结果的随机性

基于以上讨论，我认为弱监督下的 SED 的实验结果波动性非常大，以上给出的所有指标都仅供参考，在不同环境中复现时可能会有截然不同的性能表现。

另外，单纯的修改初始化的随机数种子也能给实验结果带来巨大的波动，而我认为这样的调参是毫无意义的，因此并没有进行尝试。

4. 实验总结

本次实验，我调研了声音事件检测的背景，了解了弱监督的难点；深入理解代码逻辑和训练流程，掌握 baseline 设计原理并改进 CRNN 模型；我还调研了数据增强的方式并应用，最后调整参数以达到较好地性能。在实验过程中，我发现了很多额外的问题，如训练结果的随机性大、插值函数的不稳定等等，我积极的探索其中原理，并尝试解决，在这过程中我收获颇丰。

5. References

[1] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, "Sound event detection: A tutorial," *IEEE Signal Processing Magazine*, vol. 38, no. 5, pp. 67–83, 2021.

[2] H. Dinkel, M. Wu, and K. Yu, "Towards duration robust weakly supervised sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 887–900, 2021.

[3] E. Çakir and T. Virtanen, "End-to-end polyphonic sound event detection using convolutional recurrent neural networks with learned time-frequency representation input," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–7.

[4] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *In International Symposium on Music Information Retrieval*. Citeseer, 2000.

[5] Y. Wang, J. Li, and F. Metze, "A comparison of five multiple instance learning pooling functions for sound event detection with weak labeling," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 31–35.

[6] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.