Freesound Competition,kaggle金牌第9名技术分享！？

Freesound Competition,kaggle金牌第9名技术分享！？来看看昂太客顶尖会员&曾经的老学员如何取得金牌成绩的！

英文版报告：

https://www.kaggle.com/c/freesound-audio-tagging-2019/discussion/95409#latest-564043 (https://www.kaggle.com/c/freesound-audio-tagging-2019/discussion/95409#latest-564043)

论文：

Code：https://www.kaggle.com/theoviel/9th-place-modeling-kernel (https://www.kaggle.com/theoviel/9th-place-modeling-kernel)

前言

以前我作为昂钛客ai[angtk.ai]一名学员&成员参加了一些cv类比赛，angtk社群聚焦各种算法实战项目.通过参加kaggle算法大赛提高成员&学员水平。这次我选择参加一个偏数据特征提取的比赛，来增强一下大数据的算法实战能力。

1.介绍

Freesound Tagging Competition是DCASE Workshop下的一个challenge task，具体任务和数据集描述不再赘述，我直接将我们的解决方案，比赛信息可以再https://www.kaggle.com/c/freesound-audio-tagging-2019/ (https://www.kaggle.com/c/freesound-audio-tagging-2019/) 看到
我们的这次比赛经历可以分为两个部分：
第1阶段：我们像许多参与者一样，通过试验常见的计算机视觉模型开始了比赛。频谱图的输入大小为256x512像素，输入图像沿第一维向上放大2倍。通过此设置，DenseNet模型证明了最佳性能：它们优于公共内核，Dnet121也更快了。通过在完全噪声集，spectral augmentation和MixUp上使用预训练，CV可以达到~0.85 +，并且单个fold，4fold的public LB和4个模型的融合分别是0.67-0.68，~0.70,0.717。尽管这些模型并未用于我们的提交，但这些实验为下一阶段提供了重要的见解。
到了第二阶段，Theo加入我们只好我们开始关心噪声数据的使用：组织者希望我们关注此竞赛的主要内容（没有外部数据，没有预训练模型，没有测试数据使用策略）。我们使用了两种策略：（1）预训练完整的噪声数据（2）预训练混合的curated dataset和最可靠标记的噪声数据。在这两种情况下，预训练之后都会对仅curated data进行微调。最可靠的标签是根据curated data训练的模型识别的。对于我们的最佳设置，我们有以下CV值（在第2阶段，我们使用5folds方案）：仅对curated data进行训练 - 0.858，预训练完整噪声数据 - 0.866，curated data+ 15k最佳噪声数据- 0.865，curated data+ 5k最佳噪声数据- 0.872。

	CV(*lwlrap*)
$n = 0$	0.858±0.005
$n = 5000$	**0.872**±0.005
$n = 15000$	0.865±0.004
$n = \max$	0.866±0.004
pretraining, $n = 0$	0.870±0.005
pretraining, $n = 5000$	**0.872**±0.005

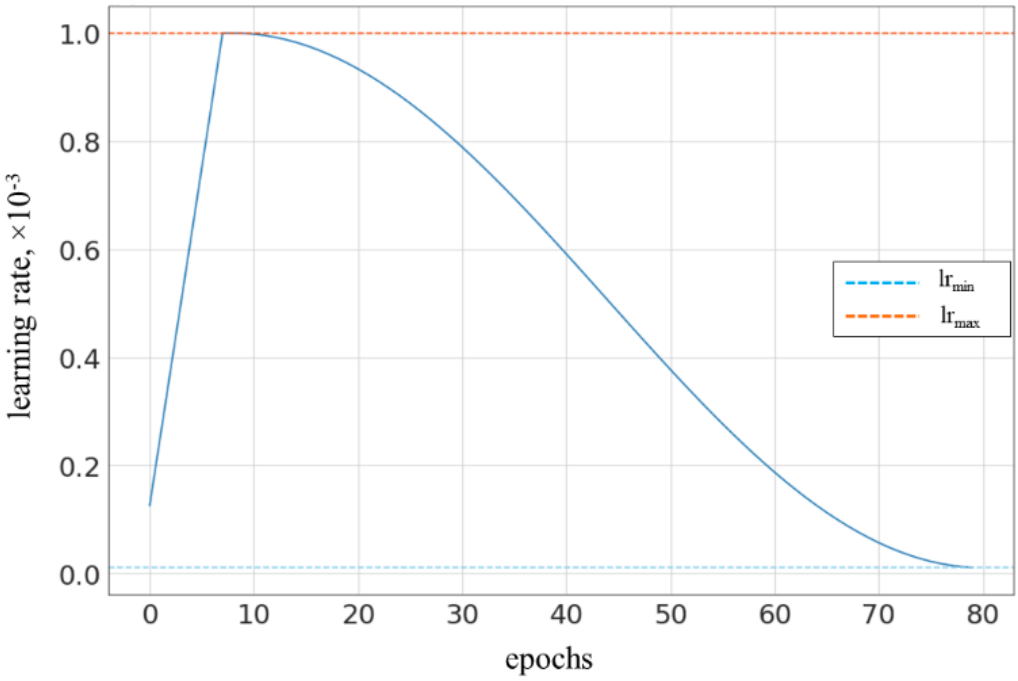
2.数据处理

Smaller and Faster makes better，我们使用64mels以及4s的intervals 在小型模型上得到了比DenseNet在128mels更高的CV以及更快的收敛速度。以至于我们能在8小时内训练完一个5folds的模型。

3.训练和预测

3.1训练步骤

在预训练阶段，我们使用一个余弦退火循环和warm up。对于不同的设置，最大lr是0.001，并且epoch的数量在50到70之间。在微调阶段，我们多次应用ReduceLROnPlateau来交替高低lr。代码是用Pytorch实现的。一次训练的总时间为1-2小时，因此整个模型的训练为6-9小时。



3.2Noisy data的处理方式

如introduction里面所说

3.3数据增强

处理音频数据的主要方法是MixUp。与真实物体的图像相比，声音是透明的：它们不会相互遮挡。因此，MixUp对音频非常有效，并提供0.01-0.015的CV提升。在第一阶段，设置MixUp的Alpha值为1.0取得了最

佳结果，而在阶段2，我们使用了0.4。 Spectral augmentation （https://arxiv.org/abs/1904.08779）（对于频率和时间具有2个masks，覆盖范围时间分别在0s和0.15s和0.3s之间）给出约0.005CV增强。我们没有在时域中使用拉伸光谱图，因为它提供了较低的模型性能。在几个模型中，我们还使用了Multisample (https://arxiv.org/abs/1904.08779）（对于频率和时间具有2个masks，覆盖范围时间分别在0s和0.15s和0.3s之间）给出约0.005CV增强。我们没有在时域中使用拉伸光谱图，因为它提供了较低的模型性能。在几个模型中，我们还使用了Multisample) Dropout（其他模型训练没有使用Dropout，希望不要收到谷歌的律师函哈哈哈哈哈哈）; 但是，它将CV降低了~0.002。我们没有应用水平翻转，因为它降低了CV并且也不自然：我不认为人们能够识别从后面播放的声音，它与使用垂直翻转训练ImageNet相同。

3.4模型架构

模型方面我们第二阶段使用了https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch (https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch) 作为基础模型，根据我们之前对DensNet的经验，我们在卷积块内部添加了dense连接并且做了一些pooling ，并在我们的最佳实验（模型M1）中连接池，将性能提升到0.868。卷积块数从4增加到5只得到0.865 CV。对于第2,3和4个卷积块（M2）使用pyramidal pooling 会产生比M1更差的结果。最后，我们的最终设置（M3）由5个卷积块 和pyramidal pooling 组成，达到了0.872 CV。同一结构的DenseNet121仅达到0.836（DenseNet121需要更高的图像分辨率，256x512，达到0.85+ CV）。从实验中，它寻找音频，在通过合并缩小尺寸之前进行非线性操作很重要，尽管我们没有详细检查它。我们使用M1，M2和M3来创建我们整体的可变性。由于提交限制，我们检查了Public LB中少数模型的性能，最佳单一模型得分为0.715。
这是M3的结构图

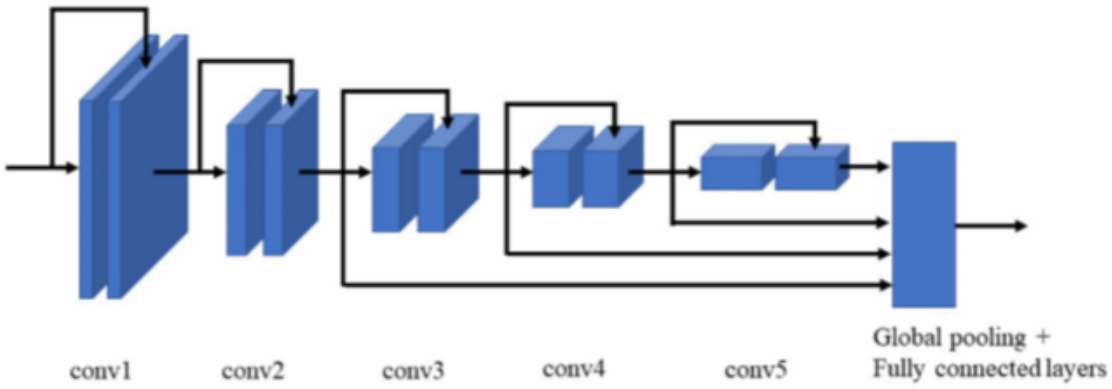


Table 2: Description of neural network architectures. Convolutional blocks are encapsulated in square brackets, and the first two numbers in each line represent the kernel size and the number of filters. Concatenate referees to concatenation of the input and the result of the first convolutional layer in a block. *Pyramid Pooling* refers to concatenation of the results of the second, third, fourth (and fifth) convolutional blocks. In the last row, numbers represent the number of output features of each linear layer.

Feature size	M0 (baseline)	M1	M2	M3	M4
64×256 × 1	Log mel-spectrograms				
64×256 × 64	$\begin{bmatrix} 3\times3, 64, \text{BN}, \text{ReLU} \\ 3\times3, 64, \text{BN}, \text{ReLU} \end{bmatrix}$			$\begin{bmatrix} 3\times3, 64, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 64, \text{BN}, \text{ReLU} \end{bmatrix}$	
	2×2 Avr Pooling				
32×128 × 128	$\begin{bmatrix} 3\times3, 128, \text{BN}, \text{ReLU} \\ 3\times3, 128, \text{BN}, \text{ReLU} \end{bmatrix}$			$\begin{bmatrix} 3\times3, 128, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 128, \text{BN}, \text{ReLU} \end{bmatrix}$	
	2×2 Avr Pooling				
16×64 × 256	$\begin{bmatrix} 3\times3, 256, \text{BN}, \text{ReLU} \\ 3\times3, 256, \text{BN}, \text{ReLU} \end{bmatrix}$			$\begin{bmatrix} 3\times3, 256, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 256, \text{BN}, \text{ReLU} \end{bmatrix}$	
	2×2 Avr Pooling				
8 ×32 × 512	$\begin{bmatrix} 3\times3, 512, \text{BN}, \text{ReLU} \\ 3\times3, 512, \text{BN}, \text{ReLU} \end{bmatrix}$			$\begin{bmatrix} 3\times3, 512, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 512, \text{BN}, \text{ReLU} \end{bmatrix}$	
	2 ×2 Avr Pooling	2 ×2 Avr Pooling	None	2×2 Avr Pooling	
4 ×16 × 1024				$\begin{bmatrix} 3\times3, 1024, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 1024, \text{BN}, \text{ReLU} \end{bmatrix}$	
	Adaptive Max Pooling 512 features	Adaptive Concat Pooling 1024 features	Adaptive Concat Pyramid Pooling 1792 features	Adaptive Concat Pyramid Pooling 3840 features	Adaptive Concat Pooling 2048 features
	BN, 128, PReLU, BN, 80		BN, 256, PReLU, BN, 80		

Table 3: The performance of the considered model architectures evaluated as average over 5-fold CV. The error represents the standard deviation of the mean.

Model	M0	M1	M2	M3	M4	DenseNet121
CV(lwlrap)	0.855±0.006	0.868 ±0.005	0.867±0.003	**0.872** ±0.005	0.865 ±0.004	0.836 ±0.005

3.5模型融合

我们的模型只需要1-1.5分钟来预测整个testset，所以我们融合了较多的模型，选择了15个差异最大的5-folds模型，将每个类的输出都归一化到0-1的范围内，再对15个模型的输出取平均值。在Public LB中达到了0.739的lwlrap。

3.6加入我们

如果想找到我们昂钛客ai[angtk.ai]社群，可以加微信号：angtkbj