数字信号处理大作业展示

苏慧学 夏纯烜 徐少轩 赵嘉浩

2024.12.23

检索任务

Mel Spec, STFT, MFCC 复现流程

➢STFT复现难点

- 一开始并未考虑padding，frames对不上
- librosa库的stft函数中有许多步骤是用于优化，需要快速定位到关键步骤

➢Mel 滤波器复现错点

- 使用的是fft的freqs与hz_points之间的差值
- librosa最后过了一个slaney的norm

Mel Spec, STFT, MFCC 复现流程

➢Mel 谱复现难点

- librosa库默认不使用htk方法（hz2mel，mel2hz）而是使用Slaney方法，需要手动指定才能与标准库对齐。
- mels = 2595.0 * np.log10(1.0 + frequencies / 700.0)
- Slaney方法
- 低频段（0-1000Hz）：使用线性映射
- 高频段（>1000Hz）：使用对数映射
- 由于方便，我们实现直接使用htk方法，而不使用Slaney方法

使用的变体

- 关于时间求导
  - 由于音频是一个时序信号，常见形式为[num_frame, *]的矩阵，这时对时序求导，便可以增加一维特征。而且这种特征是不会被MLP所学到的。在后续会以「derivatives」标注出来
- 对STFT使用线性滤波器
  - 不以人耳为基准，使用线性滤波器，加DCT

Mel Spec, STFT, MFCC 复现结果


```
Mel Spectrogram Validation Results:
--------------------------------------------------
Our shape: (216, 40)
Librosa shape: (216, 40)
Our value range: [-75.21, 1.59]
Librosa value range: [-75.20, 1.59]
Correlation with librosa: 1.0000
Mean absolute difference: 0.0076
Maximum absolute difference: 0.1088
Standard deviation of difference: 0.0124
```

```
Correlation with librosa implementation:
MFCC 0: 1.0000
MFCC 1: 1.0000
MFCC 2: 1.0000
MFCC 3: 1.0000
MFCC 4: 1.0000
MFCC 5: 1.0000
MFCC 6: 1.0000
MFCC 7: 1.0000
MFCC 8: 1.0000
MFCC 9: 1.0000
MFCC 10: 1.0000
MFCC 11: 1.0000
MFCC 12: 1.0000
```

```
STFT 形状比较：
我们的实现：(216, 1025)
Librosa实现：(216, 1025)

幅度谱相关系数：
与Librosa的相关系数：1.0000
```

评价指标介绍

$$MRR = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{r_q}$$

$$DCG@K = \sum_{i=1}^{K}\frac{rel(i)}{\log_2(i+1)}$$

$$IDCG@K = \sum_{i=1}^{K}\frac{rel_{ideal}(i)}{\log_2(i+1)}$$

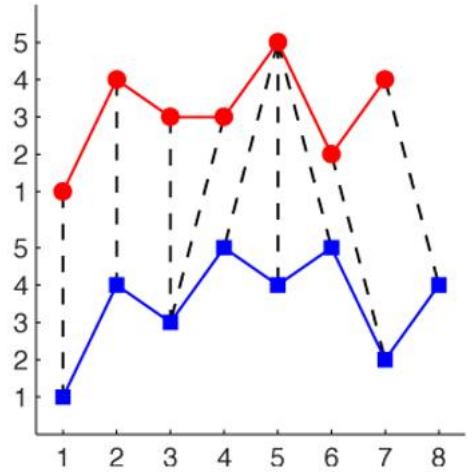
$$NDCG@K = \frac{DCG@K}{IDCG@K}$$

Binary Metric代表在前k个检索结果中，出现与query相同类别的结果就将binary置为1，否则置为0，最后取所有query的平均值

Proportion Metric代表在前k个检索结果中与query相同类别的结果占k的比例的平均值

打分模型介绍>>DTW && cos

$$DTW(i,j) = D(i,j) + \min\{\underbrace{DTW(i-1,j)}_{\text{Insertion}}, \underbrace{DTW(i,j-1)}_{Delete}, \underbrace{DTW(i-1,j-1)}_{Match}\}$$



$O(nm)$

问题在于时间复杂度过高

```python
@staticmethod
@njit(parallel=True)
def compute_dtw_distances(query_features: np.ndarray, db_features: np.ndarray) -> np.ndarray:
    n_samples = len(db_features)
    distances = np.zeros(n_samples, dtype=np.float64)

    for i in prange(n_samples):
        distances[i] = dtw_distance(query_features, db_features[i])

    return distances
```

实现层面的一些技巧，@jit！
可以把需要4h的一组实验降到14min

cos就是指简单计算了提取出的特征展开成向量之后的夹角

Autoencoder

AutoEncoder	MRR@top_k		Binary Metric		Proportion Metric	
	10	20	10	20	10	20
128 DTW	0.0764	0.0887	0.2500	0.4400	0.0310	**0.0324**
256 DTW	0.0720	0.0847	0.2600	**0.4525**	**0.0318**	0.0314
512 DTW	**0.0883**	**0.0999**	**0.2650**	0.4350	0.0315	0.0321
128 Cosine	0.0771	0.0859	0.2375	0.3650	0.0275	0.0246
256 Cosine	0.0766	0.0851	0.2325	0.3575	0.0280	0.0259
512 Cosine	0.0782	0.0893	0.2150	0.3775	0.0290	0.0293

结果展示

表 1: 使用余弦相似度检索

Cosine Similarity		MRR@top_k		NDCG@top_k		Binary Metric		Proportion Metric	
		10	20	10	20	10	20	10	20
FFT	44100,-,-	0.2784	0.2867	0.2662	0.3566	0.5350	0.6550	0.1355	0.1110
	16000,-,-	0.2639	0.2740	0.2500	0.3493	0.5125	0.6575	0.1227	0.1080
STFT	44100,4096,1024	0.2638	0.2698	0.2453	0.3166	0.4575	0.5450	0.1148	0.0934
	16000,2048,512	0.2594	0.2671	0.2391	0.3140	0.4425	0.5550	0.1123	0.0925
MFCC	44100,4096,1024	**0.3503**	**0.3590**	**0.3180**	**0.4223**	**0.6075**	**0.7300**	**0.1562**	**0.1284**
	16000,2048,512	0.3138	0.3232	0.2983	0.3942	0.5675	0.7000	0.1343	0.1099
LFCC	44100,4096,1024	0.2747	0.2840	0.2623	0.3548	0.5250	0.6625	0.1255	0.1029
	16000,2048,512	0.2899	0.2976	0.2742	0.3652	0.5475	0.6625	0.1268	0.1051
F0	44100,4096,1024	0.0904	0.0962	0.1079	0.1405	0.1950	0.2750	0.0328	0.0280
	16000,2048,512	0.1047	0.1136	0.1161	0.1672	0.2350	0.3650	0.0380	0.0360
F1	44100,4096,1024	0.0857	0.0916	0.1098	0.1453	0.2250	0.3150	0.0355	0.0296
	16000,2048,512	0.1055	0.1163	0.1238	0.1829	0.2650	0.4225	0.0410	0.0382

结果展示

表 2: 使用 DTW 检索

DTW		MRR@top_k		NDCG@top_k		Binary Metric		Proportion Metric	
		10	20	10	20	10	20	10	20
MFCC	44100,4096,1024	**0.4215**	**0.4279**	0.3739	0.4279	**0.7125**	**0.8000**	0.2020	**0.1675**
	16000,2048,512	0.4093	0.4161	**0.3749**	**0.4867**	0.7000	0.7950	**0.2035**	0.1642
LFCC	44100,4096,1024	0.2850	0.2948	0.2802	0.3753	0.5925	0.7325	0.1140	0.0943
	16000,2048,512	0.2943	0.3042	0.2855	0.3780	0.5775	0.7175	0.1190	0.0963
F0	44100,4096,1024	0.0960	0.1048	0.1116	0.1627	0.2275	0.3575	0.0368	0.0354
	16000,2048,512	0.2024	0.2155	0.2079	0.2988	0.4350	0.6275	0.0855	0.0746
F1	44100,4096,1024	0.0666	0.0765	0.0871	0.1443	0.2100	0.3625	0.0358	0.0349
	16000,2048,512	0.1553	0.1683	0.1685	0.2097	0.4125	0.6000	0.0643	0.0622

分类任务

经典机器学习方法

- 模型使用
  - 预训练的Resnet18，较为简单的CNN。适用于Mel谱等频谱图类的特征输入
  - 加入RNN序列模型，适用于STFT，MFCC等时序性较强的特征输入

机器学习分类任务

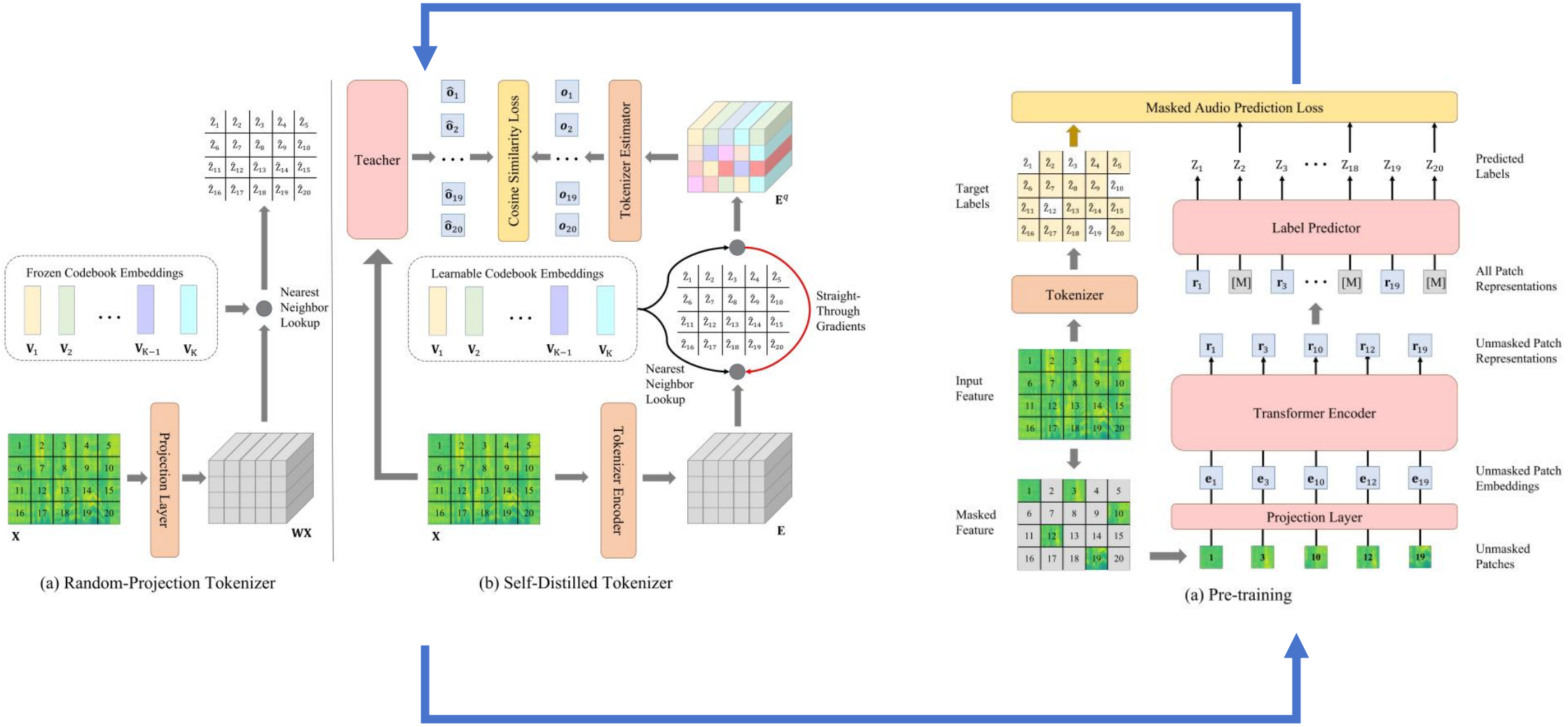
表 5: 不同特征提取方法在各类模型上的分类准确率比较 (%)

Method	Model			
	ResNet	AttentionCNN	CNN	CRNN
mel	**70.50**	51.00	38.50	23.00
mfcc_dst	55.25	37.25	37.50	18.75
mfcc_dct	57.00	40.50	34.50	19.00
mfcc_derivatives_dst	68.25	33.00	30.25	13.50
mfcc_derivatives_dct	65.75	37.75	32.00	13.75
stft	67.25	6.25	15.75	6.75
stft_dct	50.75	31.00	27.50	18.25
stft_derivatives	65.60	5.00	10.75	6.75

使用预训练且在数据集上微调的Resnet整体表现最好，尤其是在mel频谱图上。准确率达到70.5%

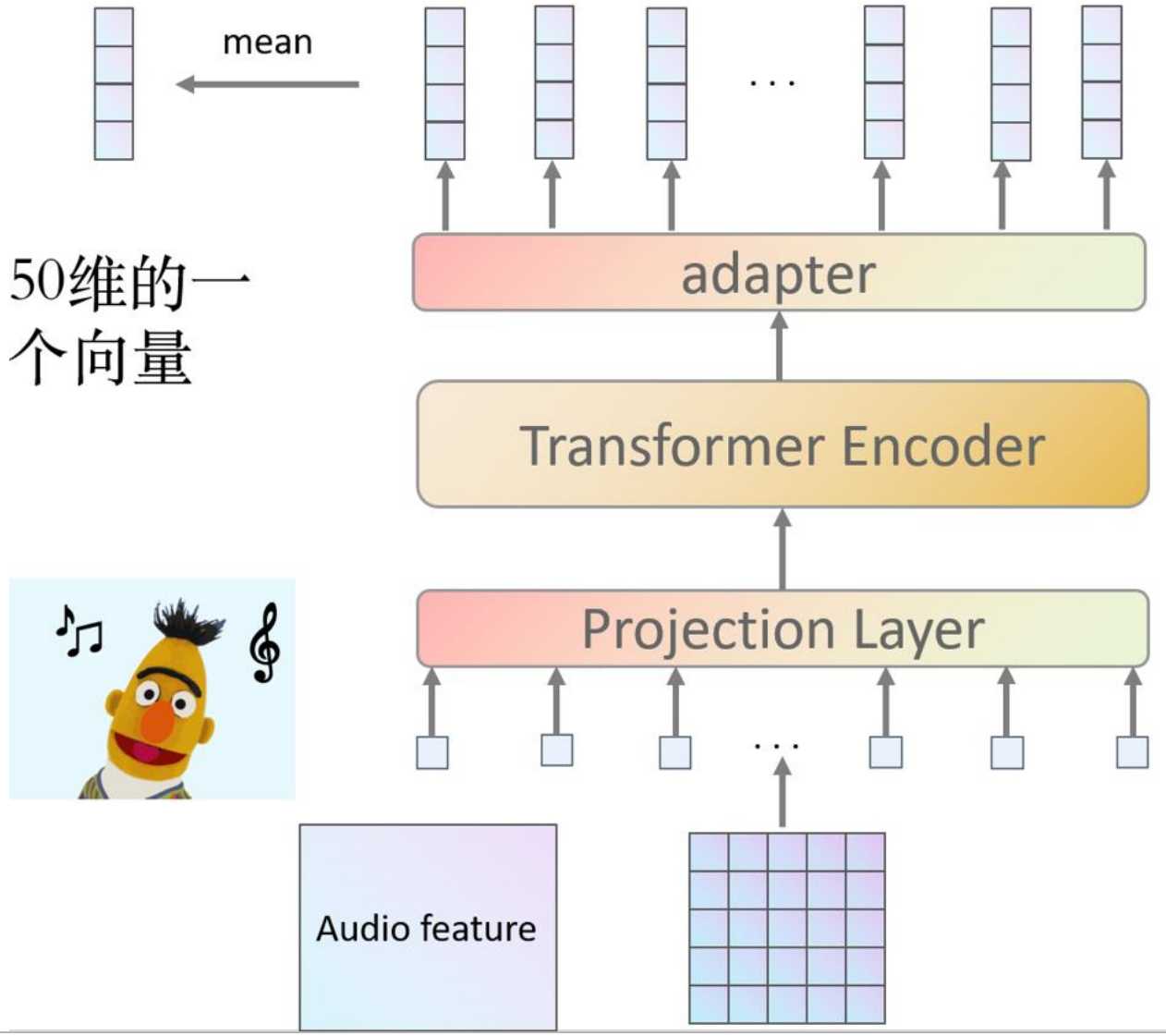
但是AttentionCNN在STFT及其导数变体这种时频域信息上表现比较差。原因可能是STFT产生的原始时频域表示包含了大量的细节信息，信息冗余。Attention在处理这种高维原始特征时也有可能难以捕捉到关键信息。

BEATS>>论文介绍

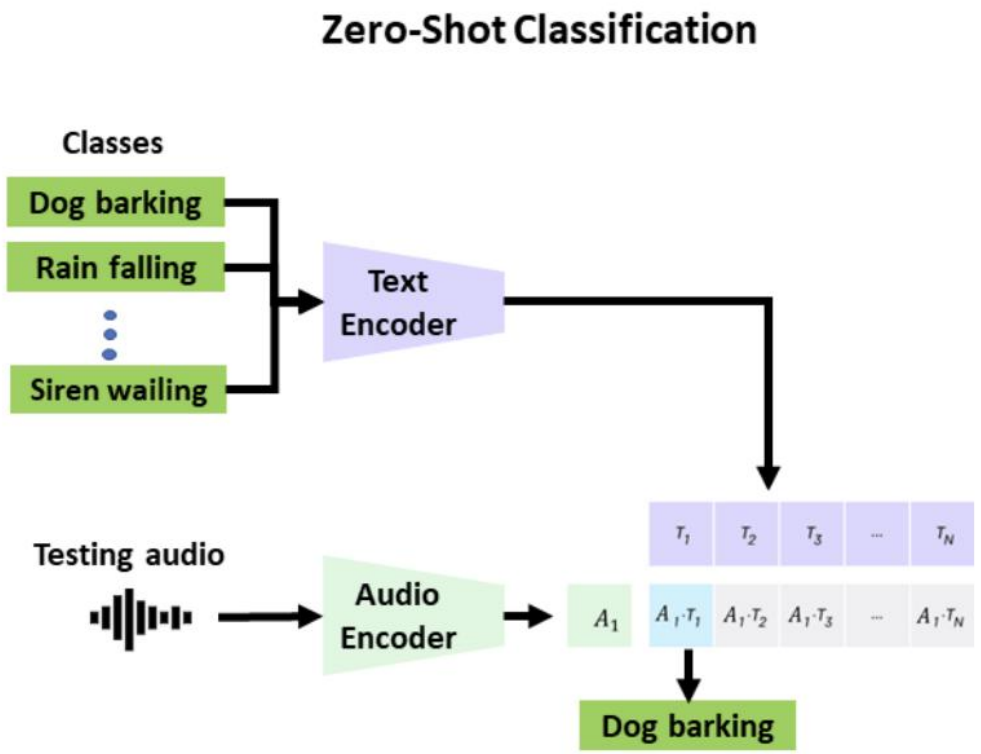

Frozen Codebook Embeddings
Nearest Neighbor Lookup
Projection Layer
X
WX
(a) Random-Projection Tokenizer
Teacher
Cosine Similarity Loss
Tokenizer Estimator
Learnable Codebook Embeddings
Straight-Through Gradients
Nearest Neighbor Lookup
Tokenizer Encoder
X
E
(b) Self-Distilled Tokenizer
Masked Audio Prediction Loss
Target Labels
Tokenizer
Input Feature
Masked Feature
Predicted Labels
Label Predictor
All Patch Representations
Unmasked Patch Representations
Transformer Encoder
Unmasked Patch Embeddings
Projection Layer
Unmasked Patches
(a) Pre-training

BEATs >>使用

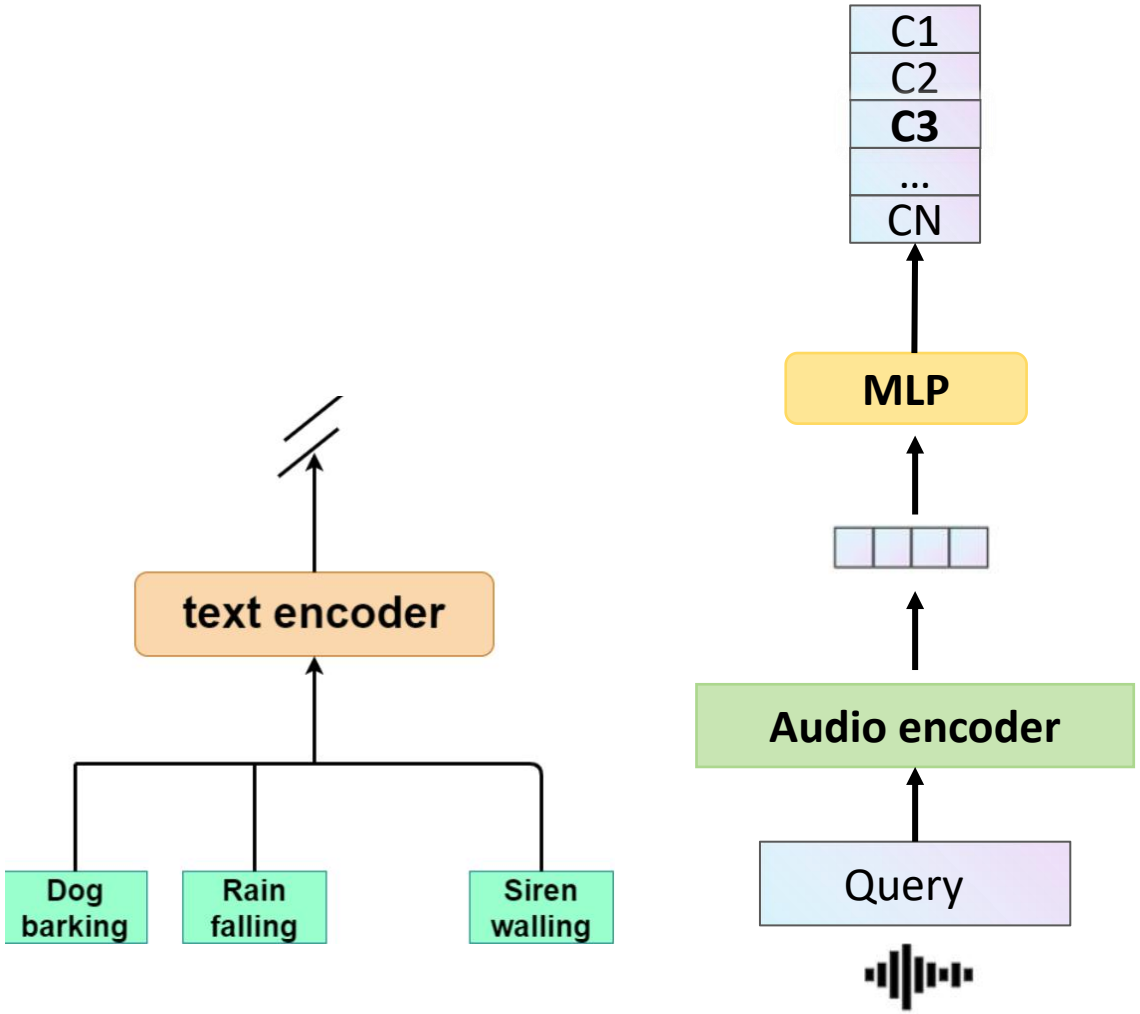
我们使用了预训练好的BEATs模型（与BERT类似），然后在ESC-50的前4个fold上用分类任务进行微调，在第五个fold上分类的acc达到了92.15%。



CLAP >>使用



全集Acc:93.85%
Fold5Acc:92.25%



Fold5Acc:98%

分类模型用于检索

Resnet分类器用于检索

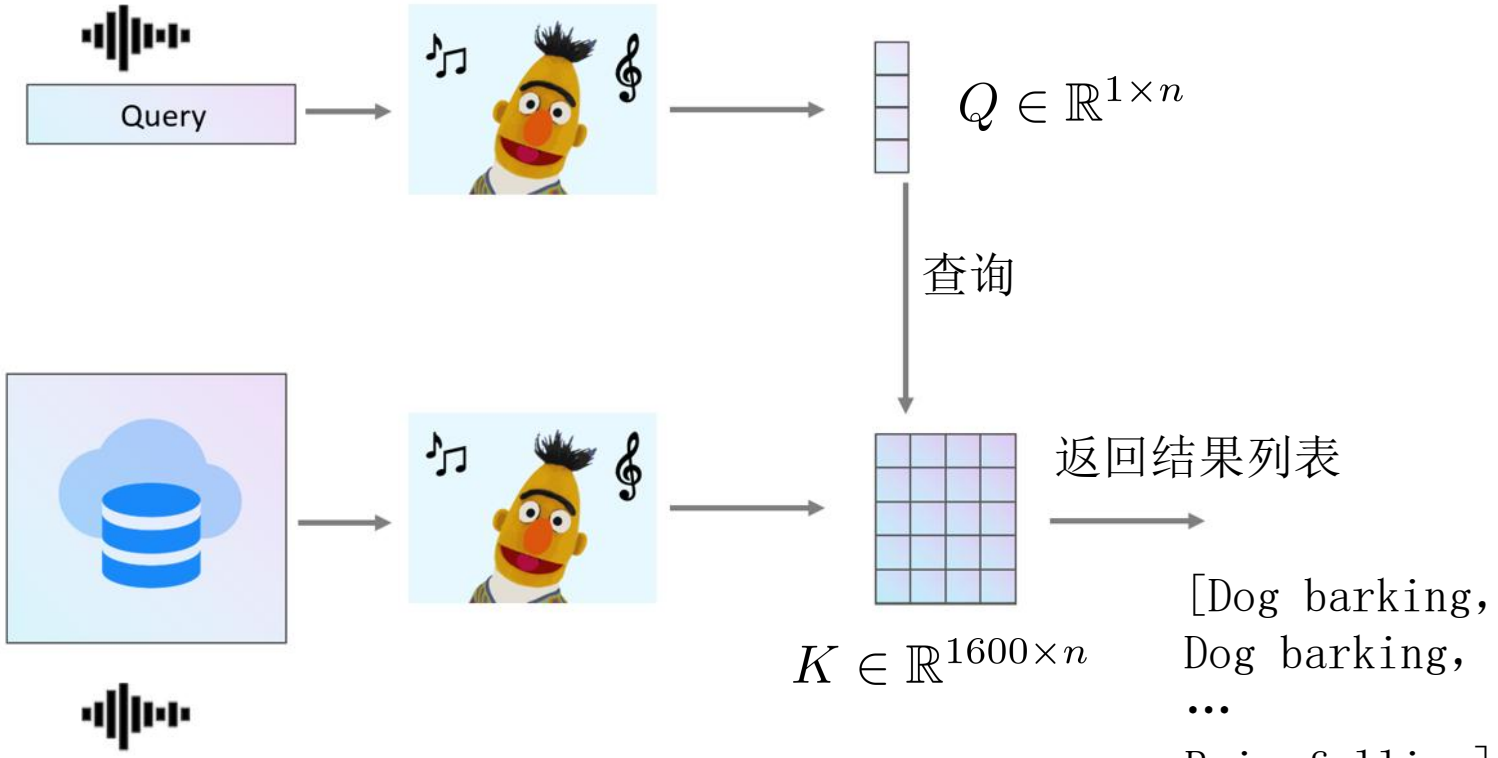
表 5: 使用 Resnet 特征向量检索

	MRR@top_k		NDCG@top_k		Binary Metric		Proportion Metric	
	10	20	10	20	10	20	10	20
mel	0.434	0.443	0.387	0.506	0.703	0.830	0.215	0.172
mfcc_dst	0.391	0.400	0.361	0.477	0.703	0.828	0.180	0.147
mfcc_dct	0.380	0.387	0.339	0.450	0.655	0.767	0.175	0.146
mfcc_derivatives_dst	0.406	0.414	0.353	0.474	0.690	0.805	0.183	0.154
mfcc_derivatives_dct	0.395	0.404	0.355	0.475	0.685	0.810	0.189	0.158
stft	**0.723**	**0.726**	**0.652**	**0.753**	**0.860**	**0.907**	**0.592**	**0.538**
stft_DCT	0.552	0.558	0.498	0.599	0.695	0.785	0.439	0.410
stft_derivatives	0.661	0.664	0.586	0.701	0.848	0.895	0.511	0.455

实现：将分类器前的隐藏向量提取出来，作为特征向量，进行向量相似度打分。

STFT方法在所有评估指标上都取得了最佳表现。但是代价是要比MFCC类和Mel谱训练起来更加费时费力。

BEATs >>使用



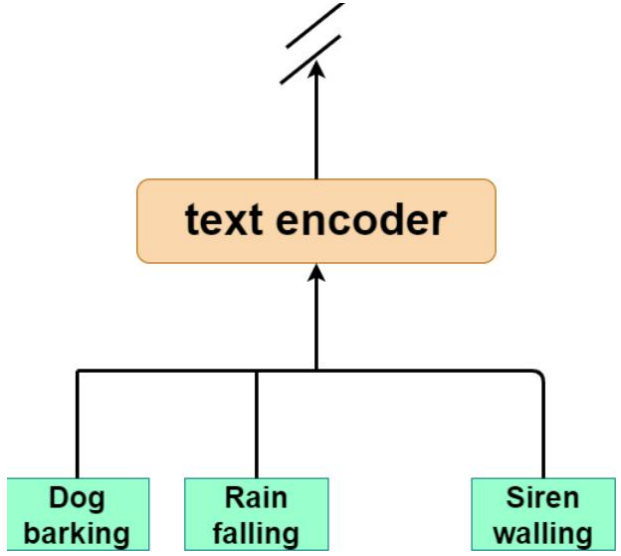
我们使用BEATs提取的特征进行检索
使用了两种提取方式

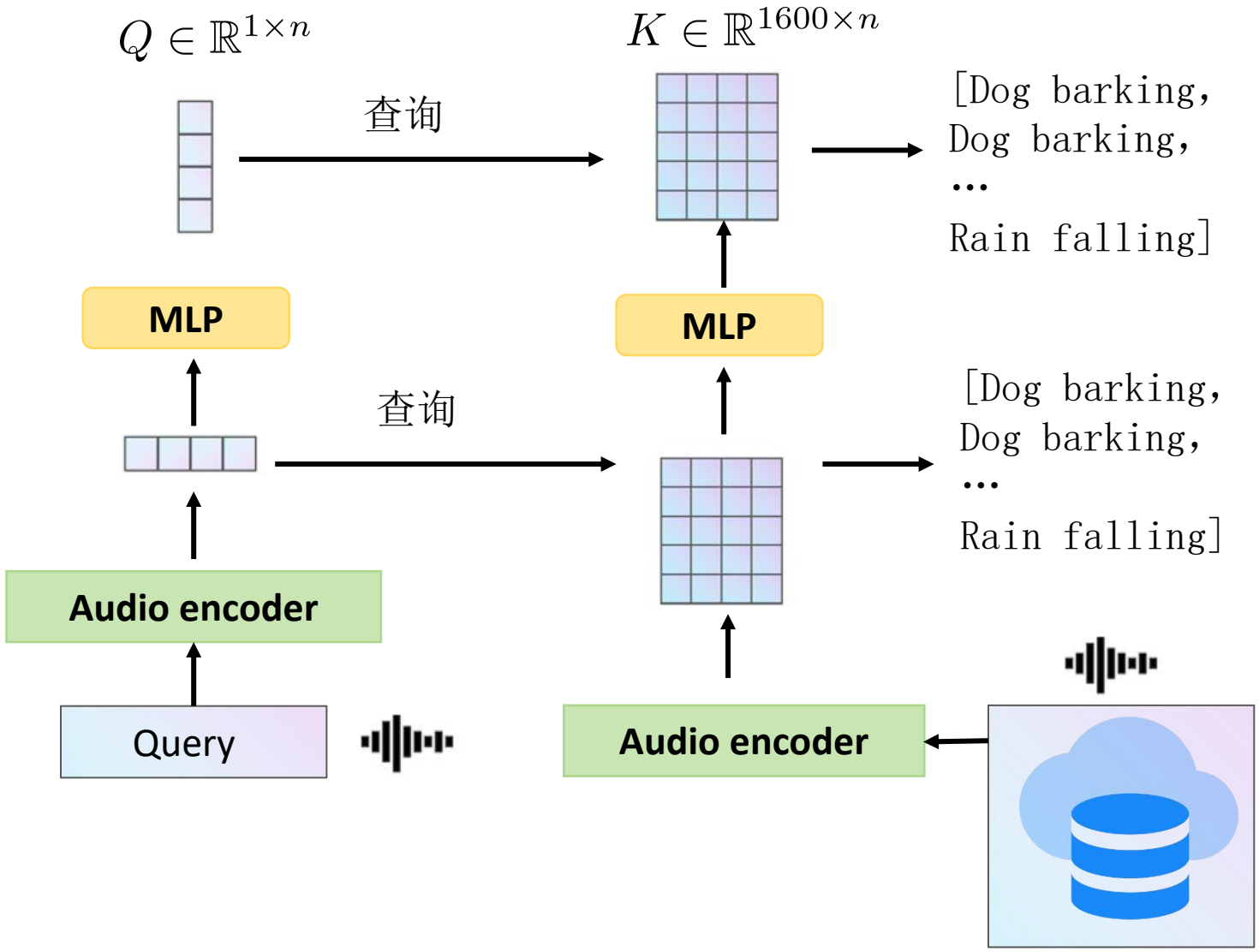
- 直接用上述微调好的BEATs输出做特征
- 用BEATs过我们加的adapter之前的输出做特征

CLAP >>使用

我们不使用text encoder，只用audio encoder在ESC-50的前4个fold上用分类任务微调，之后用双塔结构做检索。

- 直接用微调好的模型输出做特征
- 用encoder在我们加的MLP之前的输出做特征





检索结果

表 5: 使用 CLAP 及其变体检索

	MRR@top_k		NDCG@top_k		Binary Metric		Proportion Metric	
	10	20	10	20	10	20	10	20
CLAP_zero	0.9561	0.9561	0.9532	0.9598	0.9700	0.9700	0.9420	**0.9350**
CLAP_embedding	0.9637	0.9642	0.9461	0.9709	**0.9950**	**1.0000**	0.9385	0.9185
CLAP_MLP	**0.9750**	**0.9753**	**0.9565**	**0.9773**	**0.9950**	**1.0000**	**0.9478**	0.9329

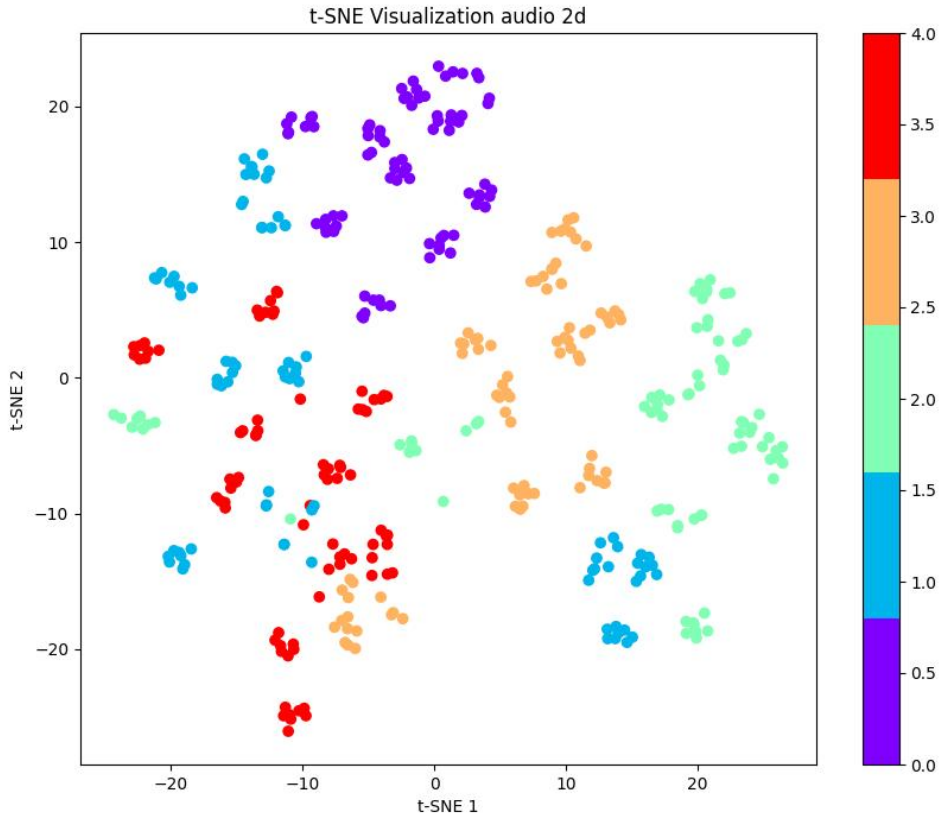
表 6: 使用 BEATs 及其变体检索

	MRR@top_k		NDCG@top_k		Binary Metric		Proportion Metric	
	10	20	10	20	10	20	10	20
BEATs_MLP	0.9215	0.9215	0.9225	0.9252	**0.9325**	**0.9325**	0.9203	0.9210
BEATs_mask_MLP	0.8192	0.8197	0.8284	0.8695	0.9125	0.9200	0.8113	0.8148
BEATs_embedding	**0.9287**	**0.9287**	**0.9264**	**0.9283**	0.9300	0.9300	**0.9235**	**0.9240**

对于CLAP，微调Audio encoder之后MRR@10可达0.975，所有query的前20个返回结果中都至少有一个同类的，并且大部分都是同类！

对于BEATs，使用微调后模型过我们添加MLP之前的特征向量来做检索，MRR@10可达0.9287，相比于无机器学习的算法取得了显著提升。

检索结果



CLAP特征按照数据集的五大类划分并可视化

原分类：keyboard typing
现分类：mouse clicking

原分类：helicopter
现分类：engine

感谢聆听！

如有不足请老师和同学们指正