mel谱图的读取、对数据集进行交叉验证分类

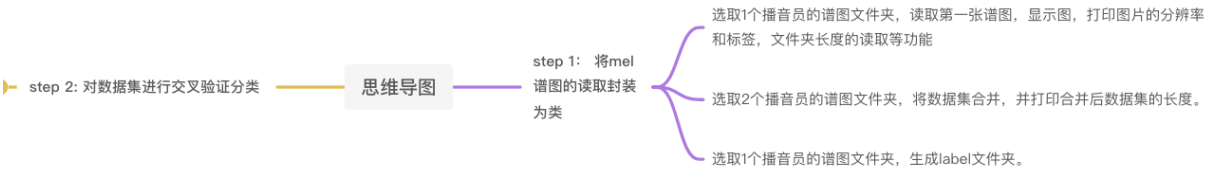
Reviewed

实验报告

实验进展：

已完成基础部分和提高部分所有实验内容

实验思维导图



实验内容：

> **1）将mel谱图的读取封装为类**
> step1: 选取1个播音员的谱图文件夹，读取第一张谱图，显示图，打印图片的分辨率和标签，文件夹长度的读取等功能
> step2: 选取2个播音员的谱图文件夹，将数据集合并，并打印合并后数据集的长度。
> step3: 选取1个播音员的谱图文件夹，生成label文件夹。
> **2）提高任务：**
> 对数据集进行交叉验证分类（可以把这个功能封装为类），输出trainset长度和validationset长度。

将mel谱图的读取封装为类

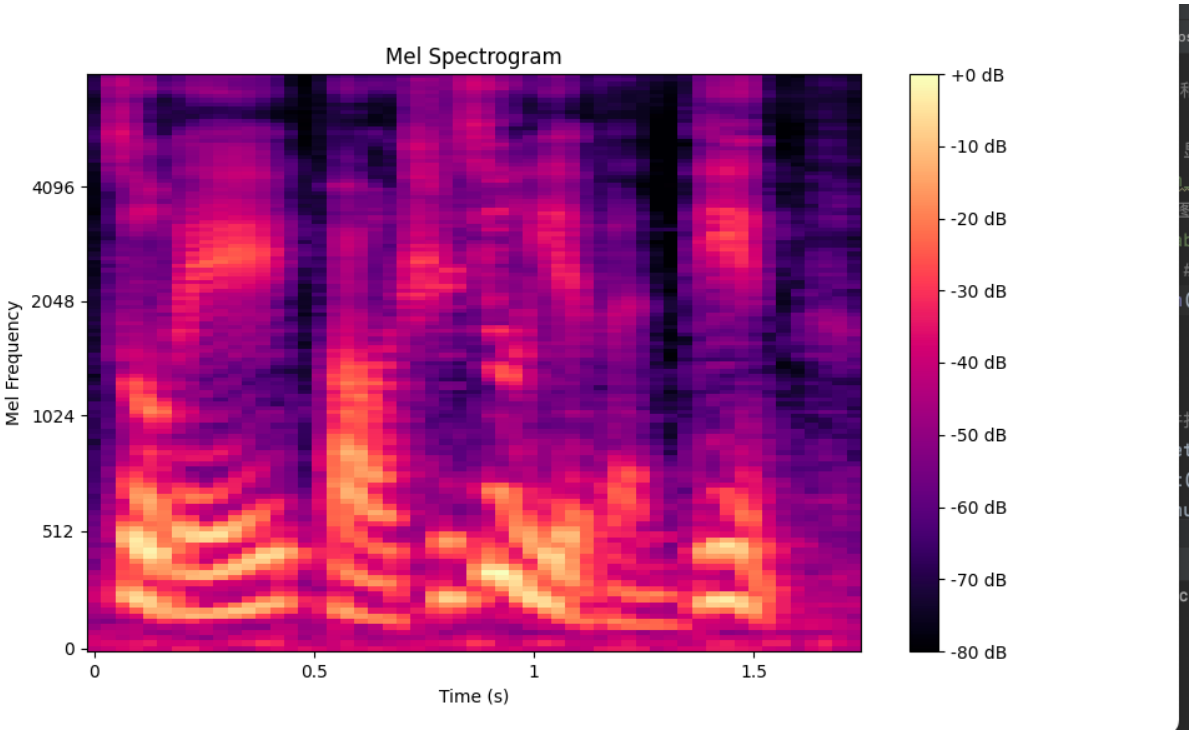
```
import os
import cv2
import numpy as np
```

```
from torch.utils.data import Dataset

class MelSpectrogramDataset(Dataset):
    '''该类封装了读取谱图的功能'''
    def __init__(self, root_dir):
        '''初始化类MelSpectrogramDataset'''
        self.root_dir = root_dir
        self.file_list = sorted(os.listdir(self.root_dir))

    def __len__(self):
        '''返回谱图文件夹中文件的数量'''
        return len(self.file_list)

    def __getitem__(self, index):
        '''根据索引读取文件并返回文件和标签。'''
        file_path = os.path.join(self.root_dir, self.file_list[index])
        label = file_path.split("/")[-1][:-4] # 文件夹名即标签，通过文件路径获取标签
        image = cv2.imread(file_path)  # 读取图片
        return image, label  # 返回图片和标签
```

1、选取1个播音员的谱图文件夹，读取第一张谱图，显示图，打印图片的分辨率和标签，文件夹长度的读取等功能

```
# 选取1个播音员的谱图文件夹，读取第一张谱图，显示图，打印图片的分辨率和标签，文件夹长度的读取
dataset = MelSpectrogramDataset("Boyin_mel_save/haixia")
image, label = dataset[0]  # 获取第一张图片和对应的标签
print(f"Image shape: {image.shape}, Label: {label}")  # 打印图片尺寸和标签
cv2.imshow("Mel Spectrogram", image)  # 展示图片
print(f"Length of haixia_dataset: {len(dataset)}")  # 打印Boyin_mel_save/haixia/文件夹的长度
# cv2.waitKey(0)  # 等待按键
cv2.destroyAllWindows()  # 关闭所有窗口
```

输出结果：




```
/Users/palekiller/opt/anaconda3/envs/YuYinShiBie/bin/p
Image shape: (600, 1000, 3), Label: haixia001_0
Length of haixia_dataset: 200
```

2、选取2个播音员的谱图文件夹，将数据集合并，并打印合并后数据集的长度。

```
# 选取2个播音员的谱图文件夹，将数据集合并，并打印合并后数据集的长度
kanghui_dataset = MelSpectrogramDataset("Boyin_mel_save/kanghui")
haixia_dataset = MelSpectrogramDataset("Boyin_mel_save/haixia")
merged_dataset = np.concatenate((kanghui_dataset, haixia_dataset))
print(f"Merged dataset length: {len(merged_dataset)}")
```

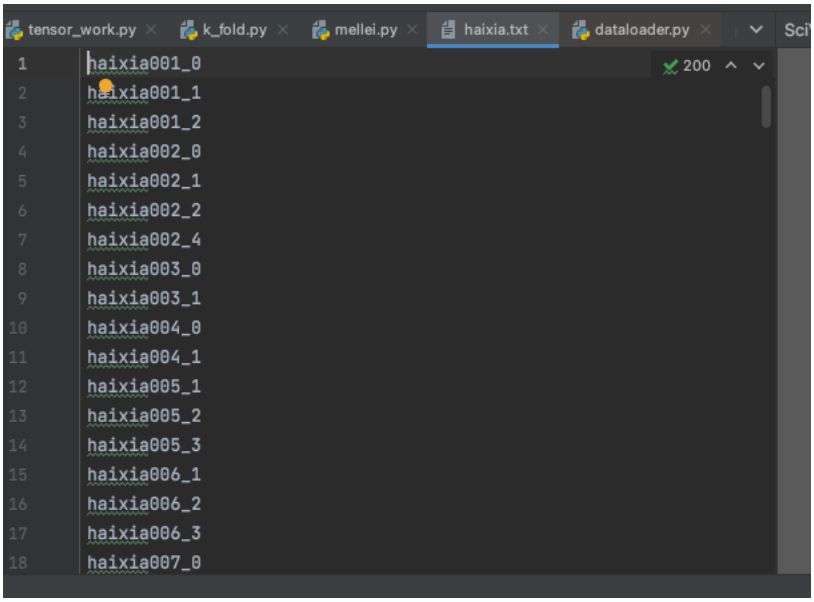
输出结果：

```
Merged dataset length: 400
```

3、选取1个播音员的谱图文件夹，生成label文件夹。

```
# 选取1个播音员的谱图文件夹，生成label文件夹
if not os.path.exists("labels"):
    os.mkdir("labels")
with open("labels/haixia.txt", "w") as f:
    for i in range(len(haixia_dataset)):
        label = haixia_dataset[i][1]
        f.write(f"{label}\n")
```

输出结果：



对数据集进行交叉验证分类（可以把这个功能封装为类），输出trainset长度和validationset长度。

```
class KFoldSplitter:
    '''
    为了实现交叉验证，我们可以先将数据集进行随机划分，
    然后每次选取其中一部分作为验证集，
    其余部分作为训练集。
```

```
    '''
    def __init__(self, dataset, num_folds=5, shuffle=True):
        self.dataset = dataset
        self.num_folds = num_folds
        self.shuffle = shuffle

    def split(self):
        indices = list(range(len(self.dataset)))
        if self.shuffle:
            random.shuffle(indices)

        fold_size = len(self.dataset) // self.num_folds
        for i in range(self.num_folds):
            start = i * fold_size
            end = (i+1) * fold_size if i < self.num_folds-1 else len(self.dataset)
            val_indices = indices[start:end]
            train_indices = list(set(indices) - set(val_indices))
            yield train_indices, val_indices


dataset = MelSpectrogramDataset("Boyin_mel_save/kanghui")  # mel谱图数据集
splitter = KFoldSplitter(dataset, num_folds=5, shuffle=True)  # 创建交叉验证实例
# 循环迭代器返回的每一个训练集和验证集的索引
for fold, (train_indices, val_indices) in enumerate(splitter.split()):
    '''
    这里使用了 splitter.split() 返回的迭代器，
    每次迭代都会返回一个元组，其中包含训练集和验证集的索引，
    enumerate() 函数用于将迭代器中的每个元素与其对应的索引进行关联，并返回一个元组。
    '''
    trainset = [dataset[i] for i in train_indices]  # 获取训练集（将训练集的索引 train_indices 与数据集实例 dataset 关联起来，获取训练集数据。）
    valset = [dataset[i] for i in val_indices]  # 获取验证集（将验证集的索引 val_indices 与数据集实例 dataset 关联起来，获取验证集数据。）
    print(f"Fold {fold+1}: trainset length: {len(trainset)}, validationset length: {len(valset)}")  # 输出当前交叉验证的折数 fold，以及训练集和验证
```

```
Fold 1: trainset length: 160, validationset length: 40
Fold 2: trainset length: 160, validationset length: 40
Fold 3: trainset length: 160, validationset length: 40
Fold 4: trainset length: 160, validationset length: 40
Fold 5: trainset length: 160, validationset length: 40

Process finished with exit code 0
```