# 水下目标识别（声纹识别）项目

本项目聚焦于水下目标识别领域，利用声纹识别技术实现对水下目标的分类与定位。通过精心设计的模型架构、数据处理流程和训练策略，旨在提高水下目标识别的准确率和效率。

## 一、项目概述

项目核心是基于深度学习的水下目标识别系统，主要包含数据处理、模型构建、训练与评估等模块。数据处理部分负责从音频文件中提取梅尔频谱、韦尔奇功率谱和平均幅度谱等声学特征，并构建数据集；模型构建涵盖多个神经网络模型，如用于分类的`class_network`、用于定位的`local_network`以及结合两者的多任务模型`MultiTaskLossWrapper`；训练与评估模块则运用优化算法和评估指标，对模型进行训练和性能评估。

## 二、项目结构

```
project/
│
├── nw_class.py          # 分类网络模型定义
├── nw_class_moe.py      # 带有MoE结构的分类网络模型定义
├── nw_local.py          # 定位网络模型定义
├── nw_local_moe.py      # 带有MoE结构的定位网络模型定义
├── nw_mtl.py            # 多任务学习模型定义
├── md_moe_rl.py         # 音频数据处理及数据集构建
├── train_mtl.py         # 模型训练与测试主程序
├── pltpdf.py            # 训练过程数据可视化
├── requirements.txt     # 项目依赖库列表
└── README.md            # 项目说明文档
├── data_gen/            # 水下声信道数据生成相关文件
│    ├── dataOcnMptEhc_Pos.m      # 海洋声信道数据生成主流程
│    ├── Pos1Azi1freq100Hz.env    # BELLHOP相关的环境参数文件
│    ├── Pos1Azi1freq100Hz.brc    # BELLHOP相关的配置文件
│    ├── Pos1Azi1freq100Hz.bty    # BELLHOP相关的海底地形数据文件
│    ├── Pos1Azi1freq100Hz.ssp    # BELLHOP相关的声速剖面数据文件
│    ├── Pos1Azi1freq100Hz.arr    # BELLHOP生成的到达数据文件
│    ├── bellhop.m                # 运行BELLHOP程序
│    ├── depd.m                   # 复制文件到当前工作目录
│    ├── funDirFolder.m           # 列出指定路径下的所有文件夹
│    ├── funNorm.m                # 对信号进行均值归一化处理
│    ├── funOME.m                 # 海洋环境多径效应建模函数
│    ├── funReadTestLb.m          # 读取测试标签文本文件
│    ├── help.md                  # 包含导出Conda环境依赖等帮助信息
│    ├── natsort.m                # 对文本数组进行自然排序
│    ├── natsortfiles.m           # 对文件名或文件夹名进行自然排序
│    └── read_arrivals_bin.m      # 读取BELLHOP生成的二进制格式的到达数据文件
```

## 三、安装指南

1. **环境准备**：确保已安装Python环境，建议使用Python 3.7及以上版本。
2. **安装依赖库**：在项目根目录下，执行以下命令安装项目所需的依赖库：

```
pip install -r requirements.txt
```

## 四、数据准备

1. 准备包含水下目标音频数据的文件，音频格式需支持`librosa`库读取，建议采样率为16kHz。
2. 数据文件应按行排列，每行格式为`音频文件路径\t标签\t距离\t深度`，如`E:/data/audio1.wav\t0\t10.0\t5.0`。
3. 将训练数据和测试数据分别整理成列表文件，如`train_list.txt`和`test_list.txt`，同时准备类别标签列表文件`label_list.txt`，每行一个标签。
4. 配置文件`config.json`需包含`Rrmax`和`Szmax`两个字段，分别用于距离和深度的归一化。

下载开源的数据集
原始的shipsear 16k 采样 5s 分割数据
https://www.doubao.com/drive/s/8623c37953c6a3d6
经过海洋声信道处理的shipsear 16k 采样 5s 分割数据
https://www.doubao.com/drive/s/4914e4ad2b3ec87a

### 使用 Matlab 脚本一键生成数据集

**1. 代码概述**

`data_gen` 文件夹中的 Matlab 代码主要用于生成水下声信道数据，模拟海洋环境中的多径效应。这些代码包括加载 BELLHOP 仿真结果、应用声场传播模型到音频数据以及生成多位置声学特征数据集等功能。

**2. 主要文件及功能**

- `dataOcnMptEhc_Pos.m` ：海洋声信道数据生成主流程，包括加载原始音频数据、应用声场传播模型、生成多位置声学特征数据集，并将结果保存到指定路径。
- `funOME.m` ：海洋环境多径效应建模函数，对输入的时域信号添加多径效应。
- `funReadTestLb.m` ：读取测试标签文本文件，返回包含每行文本的单元格数组。
- `read_arrivals_bin.m` ：读取 BELLHOP 生成的二进制格式的到达数据文件。
- `bellhop.m` ：运行 BELLHOP 程序。
- `funNorm.m` ：对信号进行均值归一化处理。
- `depd.m` ：复制文件到当前工作目录。
- `Pos1Azi1freq100Hz.env`:配置文件，包含 BELLHOP 仿真的环境参数。

**3. 使用方法**

1. 确保 Matlab 环境中已经安装了 BELLHOP 程序，并且其可执行文件 `bellhop.exe` 在 Matlab 的搜索路径中。
2. 修改 `dataOcnMptEhc_Pos.m` 文件中的 `oriDataPath` 和 `ocnDataPath` 变量，分别指定原始音频数据路径和生成数据的存储路径。
3. 运行 `dataOcnMptEhc_Pos.m` 文件，即可开始生成水下声信道数据。

## 五、模型训练

在项目根目录下，使用以下命令进行模型训练：

```
python train_mtl.py --num_epoch [训练轮数] --train_list_path [训练数据列表路径] --test_list_path [测试数据列表路径] --label_list_path [标签列表路径] --num_classes [类别数] --batch_size [训练批次大小] --test_batch [测试批次大小] --lr [学习率] --weight-decay [权重衰减系数]
```

例如：

```
python train_mtl.py --num_epoch 150 --train_list_path train_list.txt --test_list_path test_list.txt --label_list_path label_list.txt --num_classes 5 --batch_size 64 --test_batch 64 --lr 0.001 --weight-decay 5e-4
```

参数说明：

- `--num_epoch`：训练轮数，默认150。
- `--train_list_path`：训练数据列表路径。
- `--test_list_path`：测试数据列表路径。
- `--label_list_path`：标签列表路径。
- `--num_classes`：分类类别数。
- `--batch_size`：训练批次大小，默认64。
- `--test_batch`：测试批次大小，默认64。
- `--lr`：初始学习率，默认0.001。
- `--weight-decay`：权重衰减系数，默认5e-4。

训练代码可以指定输入.wav的特征，比如mel, welch, avg, gfcc, stft, cqt，在使用meg_mix作为网络时，可以使用混合输入。

网络可以从['meg', 'mcl', 'meg_blc', 'meg_mix', 'resnet18', 'resnet50', 'convnext', 'vgg16', 'vgg19', 'mobilenetv2', 'densenet121', 'swin']中选择

网络可以如下分类

- 我们的网络
  - mcl 不加混合专家模型的原始多任务分类识别网络
  - meg 加入混合专家模型的多任务、多专家、多门网络
  - meg_blc 在meg的基础上加入了负载均衡损失
  - meg_mix 不同特征 (如输入stft gfcc) 作为输入的混合专家网络
- 剩下的为常规网络

## 六、模型评估

1. 训练过程中，模型会在每个训练轮次结束后自动进行评估，并输出测试准确率、混淆矩阵、测试损失等指标。

2. 若只需进行评估，可在训练命令中添加`--evaluate`参数：

```
python train_mtl.py --evaluate --test_list_path [测试数据列表路径] --
label_list_path [标签列表路径] --num_classes [类别数] --batch_size [测试批次大小]
```

## 七、可视化分析

训练结束后，`pltpdf.py`脚本会根据训练过程中记录的数据生成训练损失、多任务权重、准确率和平均绝对误差（ABSE）等曲线，并保存为PDF文件。文件保存在模型保存路径下，可用于分析模型训练过程和性能表现。

## 八、代码示例

### 1. 音频特征提取

特征维度参数说明：

| 特征类型 | 滤波器组数量 | 特征维度 |
|---|---|---|
| Mel | 200 | [1, 200, 301] |
| STFT | - | [1, 513, 301] |
| MFCC | 40 | [1, 40, 301] |
| GFCC | 200 | [1, 200, 301] |
| CQT | 84 | [1, 84, 301] |

时间轴计算：

```
# 3秒音频总样本数 = 16000 * 3 = 48000
# hop_length=160, win_length=400
时间步数 = (48000 - 400) // 160 + 1 = 301
```

from md_moe_rl import load_audio

audio_path = "example_audio.wav"
features, welch_spectrum, avg_amp_spectrum = load_audio(audio_path)
print("梅尔频谱形状:", features.shape)
print("韦尔奇谱形状:", welch_spectrum.shape)
print("平均幅度谱形状:", avg_amp_spectrum.shape)

```
### 2. 模型调用
```python
import torch
```
```

```
from nw_mtl import MultiTaskLossWrapper

input_size = 200
channels = 512
embd_dim = 192
num_classes = 5
model = MultiTaskLossWrapper(input_size, channels, embd_dim, num_classes)

# 生成随机输入数据模拟
batch_size = 16
sequence_length = 100
input_tensor = torch.randn(batch_size, input_size, sequence_length)
welchx = torch.randn(batch_size, input_size)
avgx = torch.randn(batch_size, input_size)
label = torch.randint(0, num_classes, (batch_size,))
Rr = torch.rand(batch_size)
Sz = torch.rand(batch_size)

loss, output, outtaskLocR, outtaskLocD, prec = model(input_tensor, welchx,
avgx, label, Rr, Sz)
print("损失值:", loss.item())
print("分类输出形状:", output.shape)
print("距离定位输出形状:", outtaskLocR.shape)
print("深度定位输出形状:", outtaskLocD.shape)
print("多任务权重:", prec)
```

## 九、贡献指南

1. 欢迎广大开发者参与项目贡献，如发现问题或有改进建议，可在GitHub仓库提交Issues。
2. 若要提交代码贡献，需先fork项目仓库，在本地进行修改后，提交Pull Request，并详细描述修改内容和目的。

## 十、许可证

无

## 十一、致谢

感谢所有为项目提供帮助和支持的人员，以及相关开源库的开发者，他们的工作为项目的顺利进行提供了坚实基础。