基于对比学习与掩码重构双分支联合优化的PPG信号预训练代码

PPG-Pretrain

项目简介

PPG-Pretrain 是一个专注于 PPG (光电容积脉搏波) 信号预训练的深度学习框架，该项目结合了对比学习和掩码重构两种自监督学习任务，并通过交叉门控机制增强两种任务之间的特征交互，从而提高预训练效果。预训练的核心理念是利用大量未标记的PPG信号数据，学习通用且鲁棒的特征表示，这些预训练的特征编码器可以迁移到各种下游任务中，如内瘘狭窄程度分类任务、心率检测、睡眠阶段分类等，显著提升这些任务的性能，尤其是在标记数据有限的情况下。

核心特点

- **多任务自监督学习**：同时使用对比学习和掩码重构任务进行预训练
- **交叉门控机制**：增强两种不同自监督任务之间的特征交互
- **多尺度卷积编码**：通过不同大小的卷积核捕获PPG信号的局部和全局特征
- **层次对比损失**：在多个尺度上计算对比损失，增强模型对不同频率特征的学习
- **序列特征提取**：利用残差GRU结构提取时序依赖性

环境依赖

- Python 3.7.x　　编程语言
- PyTorch 1.12.1　　深度学习框架
- NumPy 1.21.6　　Python 科学计算的基础库，提供高效的多维数组操作
- Pandas 1.3.5　　结构化数据分析工具
- Matplotlib 3.5.3　数据可视化库
- tqdm 4.64.1　　添加进度条的库
- scikit-learn 1.0.2　　机器学习方法库

安装指南

可通过以下命令安装所有必要依赖：

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

对于特定版本的PyTorch（支持CUDA），请参考PyTorch官方安装指南。

建议使用 `conda` 创建虚拟环境并管理依赖，以避免版本冲突

文件目录结构说明

本项目包含以下核心文件：

```
PPG-Pretrain/
├── main.py                 # 主程序入口，包含训练流程
├── models.py               # 模型架构定义，包含编码器和解码器
├── data.py                 # 数据处理模块，包含数据集和增强方法
├── losses.py               # 损失函数定义，包含对比损失和重构损失
├── train.py                # 训练与验证函数，包含训练循环逻辑
└── README.md               # 项目说明文档
```

`data.py`

数据处理相关类和函数：

- **PPGDataset**：PyTorch数据集类，负责加载CSV文件中的PPG信号
- **MaskGenerator**：生成随机连续掩码
- **DataAugmenter**：实现PPG信号增强，包括添加高斯噪声、基线漂移、随机缩放等

`losses.py`

损失函数实现：

- **hierarchical_contrastive_loss**：多尺度层次对比损失
- **instance_contrastive_loss**：实例间对比损失
- **temporal_contrastive_loss**：时间维度对比损失
- **masked_reconstruction_loss**：掩码重构损失

`models.py`

模型架构定义：

- **ResidualGRU**：具有残差连接的GRU网络模块
- **Encoder**：PPG信号编码器
- **Decoder**：PPG信号解码器
- **CrossGatingModule**：交叉门控模块
- **PPGPretrainModel**：完整的预训练模型

`train.py`

训练与验证函数：

- **set_seed**：设置随机种子确保结果可复现
- **visualize_reconstruction**：可视化重构结果
- **train**：epoch训练函数
- **validate**：验证函数

`main.py`

主程序入口：

- 参数配置
- 数据加载与分割
- 模型初始化
- 训练循环
- 模型保存
- 结果可视化

数据准备

本项目使用CSV格式的PPG信号数据进行训练。PPG信号通常是从血氧仪、智能手表或其他可穿戴设备采集的。

数据格式要求

1. **文件格式**：CSV文件，每行表示一个信号样本
2. **样本长度**：每个样本应为固定长度的序列（默认600点）
3. **预处理**：信号应进行标准化或归一化处理

数据组织方式

1. 将预处理后的CSV文件放入同一目录
2. 每个CSV文件可以包含多个样本（多行）
3. 目录路径将在训练时指定

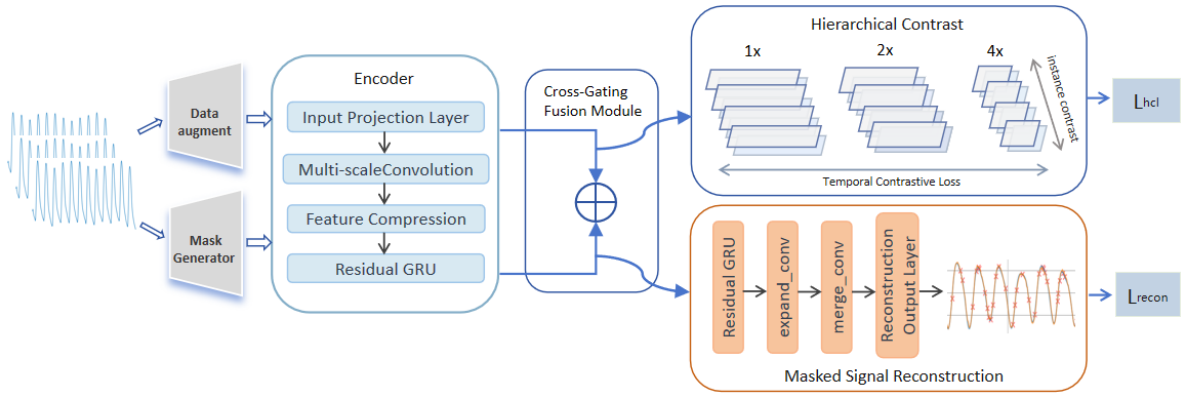
```
/path/to/data/
├── subject1.csv
├── subject2.csv
├── subject3.csv
└── ...
```

数据预处理建议

- 移除基线漂移
- 标准化信号（均值为0，标准差为1）
- 滤除异常值和噪声
- 确保所有信号长度一致（截断或填充）

模型架构

PPG-Pretrain模型由几个关键组件构成，下面是详细介绍：

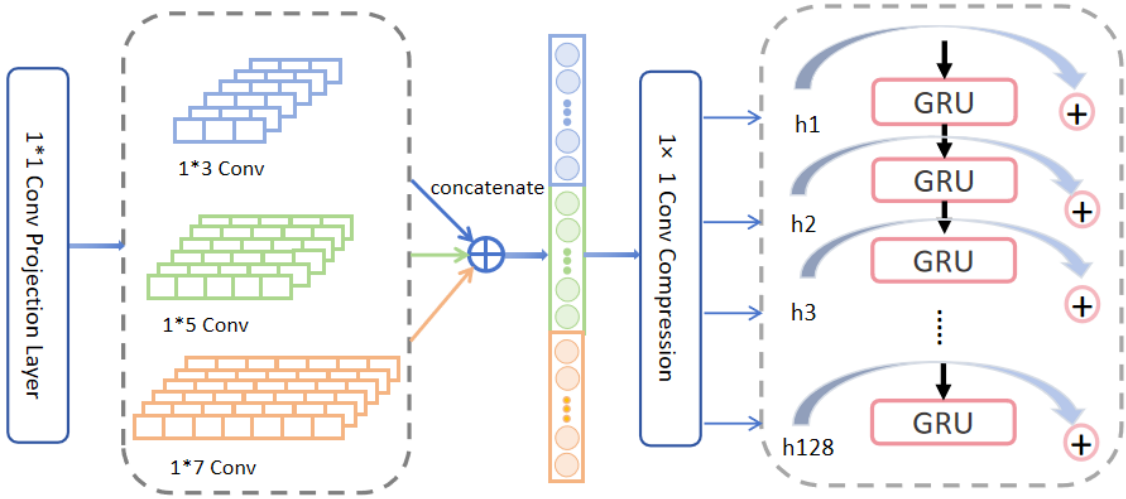


1. 编码器 (Encoder)

编码器负责将原始PPG信号转换为潜在特征表示，具有以下结构：

- **投影层**：1x1卷积，将单通道信号映射到32通道
- **多尺度卷积模块**：
  - 小尺度卷积（3x1）：捕获局部细节特征
  - 中尺度卷积（5x1）：捕获中等范围特征
  - 大尺度卷积（7x1）：捕获长程依赖特征

- **特征压缩**：将多尺度特征（192通道）压缩至128通道
- **残差GRU**：双向GRU网络，具有残差连接，提取时序特征



2. 解码器 (Decoder)

解码器负责从潜在特征重构原始信号：

- **GRU解码器**：将特征序列转换回时域
- **特征扩展**：将隐藏状态扩展到更高维度
- **多尺度转置卷积**：通过不同尺度的转置卷积重构信号
- **合并层**：将不同尺度的重构结果融合
- **输出层**：生成最终的重构信号

3. 交叉门控模块 (CrossGatingModule)

交叉门控模块是一个创新设计，用于增强对比学习特征和重构任务特征之间的交互：

- 对两个任务的特征分别进行非线性变换
- 从对方任务的特征生成门控信号
- 通过门控机制增强特征表示

4. 掩码生成器 (MaskGenerator)

生成随机连续掩码，用于掩码重构任务：

- 根据指定的掩码率确定需要掩码的点数
- 生成随机长度的连续掩码段
- 保持掩码分布满足指定的统计特性

5.数据增强 (Data Augment)

对每个原始PPG信号生成两个不同的增强视图

- 进行噪声增强、基线漂移，轻微缩放

完整模型流程

1. 数据增强生成正样本对，输入编码器
2. 生成随机掩码，掩蔽部分信号
3. 编码器提取原始信号和掩码信号的特征
4. 交叉门控模块增强两种任务的特征

5. 解码器重构被掩蔽的信号部分
6. 计算对比损失和重构损失
7. 联合优化两个任务的损失

训练流程

PPG-Pretrain采用了双分支训练策略，将对比学习和掩码重构两条路径有机结合，通过交叉门控机制增强特征表示。以下是详细的训练流程说明：

参数配置

在`main.py`中可以配置以下训练参数：

```
# 数据参数
data_dir = "/path/to/your/data"  # 数据目录路径
signal_length = 600              # PPG信号长度

# 模型参数
feature_dim = 256                # 特征维度
mask_ratio = 0.30                # 掩码比例

# 训练参数
batch_size = 64                  # 批次大小
epochs = 50                      # 训练轮数
learning_rate = 1e-3             # 学习率

# 损失权重
lambda_contrast = 1.0            # 对比损失权重
lambda_recon = 1.0               # 重构损失权重

# 保存参数
save_dir = 'path/to/save/models' # 模型保存路径
```

训练命令

训练模型只需运行：

```
python main.py
```

双分支训练流程详解

PPG-Pretrain的训练过程包含两个并行的自监督学习分支，每个分支专注于不同的学习目标：

1. 对比学习分支

对比学习分支通过学习相似样本间的关系，构建具有判别能力的特征表示：

1. **数据增强**：对每个原始PPG信号生成两个不同的增强视图
    - **高斯噪声增强**：添加轻微高斯噪声，模拟采集噪声
    - **基线漂移增强**：添加低频正弦波，模拟呼吸和体位变化影响
    - **随机缩放增强**：对信号幅度进行轻微缩放，模拟不同采集设备差异
2. **特征提取**：
    - 两个增强视图通过共享参数的编码器进行特征提取
    - 编码器捕获PPG信号的多尺度时序特征

3. **对比损失计算**：
    - 在多个时间尺度上计算样本间对比损失与时间对比损失
    - 拉近同一信号不同视图的特征距离，推远不同信号的特征距离
    - 建立时间维度上的一致性关系

对比学习分支的优势在于学习具有强判别能力的特征表示，对下游分类任务特别有效。

2. 掩码重构分支

掩码重构分支通过预测缺失部分，学习信号的内在结构和时序特性：

1. **掩码生成**：
    - 根据设定的掩码率（默认30%）随机生成连续的掩码段
    - 掩码段长度服从正态分布，平均长度可配置
    - 将被掩码位置的信号值置为0，模拟信号缺失
2. **编码与解码**：
    - 掩码信号通过编码器提取特征
    - 解码器尝试从编码特征中重构原始信号，特别是被掩码的部分
3. **重构损失计算**：
    - 只计算被掩码部分的重构误差
    - 使用均方误差(MSE)作为损失函数
    - 鼓励模型学习信号的内在结构和时序依赖关系

掩码重构分支擅长捕获信号的结构化信息和上下文依赖，对下游的信号重建和异常检测任务有较好效果。

3. 交叉门控机制

两个分支之间通过创新的交叉门控机制进行特征交互和增强：

1. **特征池化**：
    - 对两个分支的序列特征进行平均池化，得到全局特征向量
    - 为每个样本生成紧凑的特征表示
2. **交叉特征变换**：
    - 对两个分支的特征分别进行非线性变换
    - 从对方任务的特征生成门控信号
    - 应用门控机制增强各自的特征表示
3. **特征扩展**：
    - 将增强后的特征向量扩展回序列维度
    - 保持时序信息的完整性

交叉门控机制使两个分支能够互相借鉴学习到的不同类型特征，形成更全面的特征表示。

联合优化过程

1. **数据加载与预处理**：
    - 加载PPG信号数据集并归一化
    - 将数据分为训练集(90%)和验证集(10%)
    - 为每个批次创建数据增强和掩码操作
2. **前向传播**：
    - 分别计算对比学习分支和掩码重构分支的特征和损失
    - 应用交叉门控机制增强特征表示
    - 计算最终的联合损失

3. **反向传播与优化**：
    - 根据加权联合损失进行梯度计算
    - 更新共享编码器、解码器和交叉门控模块的参数
    - 应用学习率调度策略
4. **验证与模型选择**：
    - 定期在验证集上评估模型性能
    - 保存验证损失最低的模型
    - 可视化分析重构质量和损失曲线
5. **学习率自适应调整**：
    - 当验证损失在连续5个epoch内没有下降时，将学习率降低为原来的一半(factor=0.5)
    - 通过动态调整学习率，帮助模型跳出局部最优并改善收敛性能
    - 实现代码:

```python
scheduler = optim.lr_scheduler.ReduceLROnPlateau(
    optimizer, mode='min', factor=0.5, patience=5, verbose=True
)

# 在每个epoch后根据验证损失更新学习率
scheduler.step(val_loss)
```

训练过程监控

训练过程中会实时输出以下信息：

```
Epoch 10/50
Batch 100/150 | Contrast: 0.7823 | Recon: 0.1254
```

训练输出

训练过程会生成以下输出：

- **模型文件**：
    - `*_best.pth`：最佳模型权重
    - `*_best_full.pth`：最佳完整模型
    - `*_encoder_best.pth`：最佳编码器权重
    - `*_epoch{N}.pth`：每N个epoch的checkpoint
- **训练记录**：
    - `*_config.json`：训练配置
    - `*_history.json`：训练历史数据
    - `*_training_history.png`：损失曲线图
- **可视化结果**：
    - 重构信号与原始信号对比图
    - 掩码位置可视化

注意：时间戳格式为 `YYYYMMDD_HHMMSS`，确保每次训练的输出不会相互覆盖。

损失函数

PPG-Pretrain使用两种主要的损失函数：

1. 层次对比损失 (Hierarchical Contrastive Loss)

对比学习是一种自监督学习方法，旨在学习能区分相似和不相似样本的特征表示。在PPG-Pretrain中，我们采用了层次对比损失，这是一种多尺度的对比学习策略，特别适合时间序列数据如PPG信号。

层次对比损失的核心思想是在不同的时间尺度上同时进行对比学习，包括：

- **原始尺度**：直接对完整特征序列计算对比损失
- **下采样1次**：对特征进行2倍下采样后计算对比损失
- **下采样2次**：对特征进行4倍下采样后计算对比损失

这种多尺度策略使模型能够同时学习短期和长期的时间依赖关系。

每个尺度上的对比损失由两部分组成：

1.1 样本间对比损失 (Instance Contrastive Loss)

样本间对比损失关注的是批次中不同样本之间的对比关系：

- **正样本对**：同一原始信号的两个不同增强版本
- **负样本对**：来自不同原始信号的增强版本

数学表示如下：

```
样本间对比损失的目标是将同一信号的不同视角（正样本对）在特征空间中拉近，
同时将不同信号（负样本对）在特征空间中推远。

对于每个PPG信号的两个增强版本z1和z2，正样本对的相似度是通过点积计算的：
sim(z1_i, z2_i)

通过应用InfoNCE损失函数，我们最大化正样本对的相似度，同时最小化负样本对的相似度。
```

1.2 时间对比损失 (Temporal Contrastive Loss)

时间对比损失专注于序列内部，在时间维度上进行对比学习：

- **正样本对**：同一时间点在两个视图中的特征
- **负样本对**：不同时间点的特征

这种时间维度上的对比学习对于捕获PPG信号中的周期性和时间依赖性特别重要。

对比损失的优势在于：

- 不需要标记数据就能学习有意义的特征表示
- 增强了模型对噪声和变形的鲁棒性
- 提高了下游任务的性能和泛化能力

在代码实现中，我们采用了以下核心函数：

```python
def hierarchical_contrastive_loss(z1, z2, alpha=0.5, temporal_unit=0):
    """
    计算不同尺度上的对比损失

    参数:
    - z1, z2: 两个视图的特征 [batch_size, seq_len, feature_dim]
    - alpha: 样本间损失和时间损失的平衡系数
    - temporal_unit: 从哪个尺度开始计算时间损失

    返回:
```

```
    - total_loss: 所有尺度上对比损失的平均值
    """
    # 原始尺度、下采样1次、下采样2次三个尺度的实现
    # ...
```

2. 掩码重构损失 (Masked Reconstruction Loss)

掩码重构损失是另一种自监督学习策略，受到BERT和MAE等模型的启发。它的工作原理是：

- 随机掩盖输入PPG信号的一部分
- 训练模型预测被掩盖的部分
- 只计算被掩盖部分的重构误差

掩码重构损失使用均方误差(MSE)衡量重构信号与原始信号之间的差异：

```
def masked_reconstruction_loss(original, reconstructed, mask,
reduction='mean'):
    """
    只计算被掩码部分的重构损失

    参数:
        original: 原始信号
        reconstructed: 重构信号
        mask: 掩码，True表示被掩蔽的位置
        reduction: 损失归约方式

    返回:
        mse_loss: 被掩码部分的MSE损失
    """
    mse_loss = F.mse_loss(
        reconstructed[mask],
        original[mask],
        reduction=reduction
    )
    return mse_loss
```

3. 联合优化目标

PPG-Pretrain通过加权结合这两种损失函数，形成联合优化目标：

```
total_loss = lambda_contrast * contrast_loss + lambda_recon * recon_loss
```

这种多任务学习策略使模型能够学习更全面和鲁棒的特征表示，同时支持多种下游任务的迁移学习。

训练后目录结构

成功训练PPG-Pretrain模型后，您将获得一个结构化的输出目录，包含模型权重、训练记录和可视化结果。以下是完整的目录结构及详细说明：

```
项目根目录/
├── data.py                          # 数据处理模块
├── losses.py                        # 损失函数定义
├── main.py                          # 主程序入口
├── models.py                        # 模型架构定义
├── train.py                         # 训练与验证函数
```

```
├── README.md                          # 项目说明文档
|
├── cyz/bloodpressure/ppg_pretrain_models/  # 默认模型保存目录
|   ├── ppg_pretrain_[timestamp]_best.pth           # 验证集上性能最佳的模型权重
|   ├── ppg_pretrain_[timestamp]_best_full.pth      # 最佳完整模型（包含优化器状态等）
|   ├── ppg_pretrain_[timestamp]_config.json        # 训练配置记录
|   ├── ppg_pretrain_[timestamp]_encoder_best.pth   # 最佳编码器权重（用于下游任务）
|   ├── ppg_pretrain_[timestamp]_encoder_final.pth  # 最终编码器权重
|   ├── ppg_pretrain_[timestamp]_epoch5.pth         # 第5个epoch的checkpoint
|   ├── ppg_pretrain_[timestamp]_epoch10.pth        # 第10个epoch的checkpoint
|   ├── ppg_pretrain_[timestamp]_epoch15.pth        # 第15个epoch的checkpoint
|   ├── ...
|   ├── ppg_pretrain_[timestamp]_epoch50.pth        # 最后一个epoch的checkpoint
|   ├── ppg_pretrain_[timestamp]_final.pth          # 最终模型权重
|   ├── ppg_pretrain_[timestamp]_final_full.pth     # 最终完整模型
|   ├── ppg_pretrain_[timestamp]_history.json       # 详细训练历史记录（所有epoch的损失值）
|   └── ppg_pretrain_[timestamp]_training_history.png  # 训练过程损失曲线可视化
|
└── visualizations/                    # 重构结果可视化目录
    ├── epoch1_batch100.png            # 第1个epoch第100个batch的重构结果
    ├── epoch1_batch200.png            # 第1个epoch第200个batch的重构结果
    ├── epoch1_batch300.png            # 第1个epoch第300个batch的重构结果
    ├── ...
    ├── epoch2_batch100.png            # 第2个epoch第100个batch的重构结果
    ├── ...
    └── epoch50_batch[N].png           # 最后一个epoch的重构结果
```

文件说明

模型文件

- `*_best.pth`：在验证集上表现最佳的模型权重，通常用于下游任务微调或推理
- `*_best_full.pth`：最佳模型的完整状态，包含模型权重、优化器状态、学习率等，用于恢复训练
- `*_encoder_best.pth`：最佳模型的编码器部分，专为下游任务提取特征设计
- `*_epoch{N}.pth`：每隔5个epoch保存的检查点，可以从中断点恢复训练
- `*_final.pth`：最后一个epoch的模型权重
- `*_final_full.pth`：最后一个epoch的完整模型状态

训练记录

- `*_config.json`：训练参数配置，包括数据路径、批次大小、学习率、特征维度等
- `*_history.json`：详细记录每个epoch的训练和验证损失，包括对比损失和重构损失
- `*_training_history.png`：损失曲线图，直观展示训练过程中各损失的变化趋势

可视化结果

- `epoch{N}_batch{M}.png`：每隔一定批次保存的重构可视化结果，包含三部分：
  - 原始PPG信号波形
  - 带掩码的PPG信号（红点标记掩码位置）
  - 重构信号与原始信号的对比图

注意：时间戳格式为 `YYYYMMDD_HHMMSS`，确保每次训练的输出不会相互覆盖。