CHILD MIND INSTITUTE · FEATURED CODE COMPETITION · 10 DAYS AGO

Late Submission

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

CMI赛后讲解

2025.09

Competition Host

Child Mind Institute

Prizes & Awards

$50,000

Awards Points & Medals

Participation

11,922 Entrants

3,178 Participants

2,657 Teams

75,777 Submissions

Tags

Health

Time Series Analysis

Custom Metric

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

■ 如何上手比赛

- 比赛基础
  - Python基础：基础语法
  - 开源包使用：pandas、numpy、matplotlib
  - 机器学习包：scikit-learn
  - 深度学习包：tensorflow、pytorch

- 任务拆解
  - 题目分析：充分理解赛题含义和比赛评价目标
  - 熟悉数据：理解数据的特征和含义，对数据进行统计学分析
  - 特征工程：充分理解数据后，构造特征，为下一步建模做准备
  - 模型建立：建立机器学习或深度学习模型

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

■ 比赛描述

这项竞赛的目标是根据传感器数据判断手势。

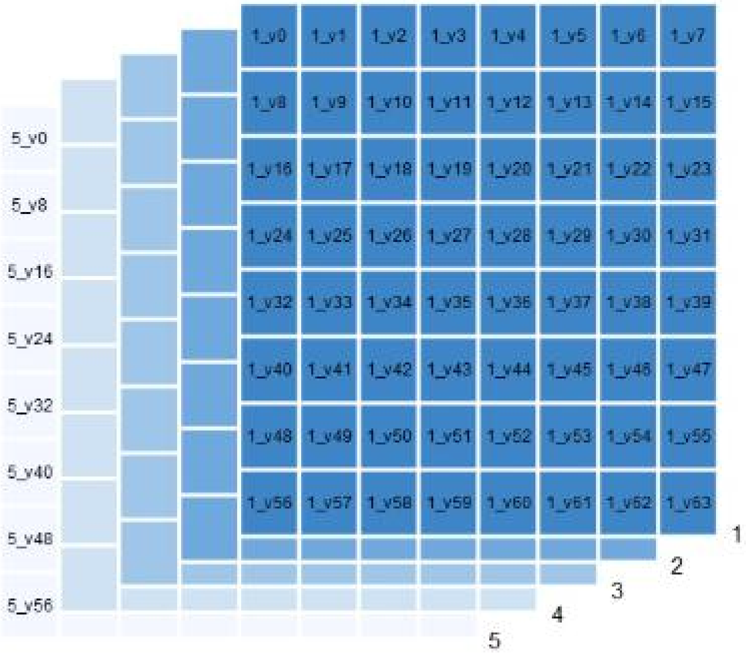
分类任务：18个label

Forehead - pull hairline	640
Neck - pinch skin	640
Text on phone	640
Neck - scratch	640
Forehead - scratch	640
Eyelash - pull hair	640
Above ear - pull hair	638
Eyebrow - pull hair	638
Cheek - pinch skin	637
Wave hello	478
Write name in air	477
Pull air toward your face	477
Feel around in tray and pull out an object	161
Write name on leg	161
Pinch knee/leg skin	161
Scratch knee/leg skin	161
Drink from bottle/cup	161
Glasses on/off	161

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

■ 数据概览



[train/test].csv

- `row_id`
- `sequence_id` - An ID for the batch of sensor data. Each sequence includes one Transition, one Pause, and one Gesture.
- `sequence_type` - If the gesture is a target or non-target type. Train only.
- `sequence_counter` - A counter of the row within each sequence.
- `subject` - A unique ID for the subject who provided the data.
- `gesture` - The target column. Description of sequence Gesture. Train only.
- `orientation` - Description of the subject's orientation during the sequence. Train only.
- `behavior` - A description of the subject's behavior during the current phase of the sequence.
- `acc_[x/y/z]` - Measure linear acceleration along three axes in meters per second squared from the IMU sensor.
- `rot_[w/x/y/z]` - Orientation data which combines information from the IMU's gyroscope, accelerometer, and magnetometer to describe the device's orientation in 3D space.
- `thm_[1-5]` - There are five thermopile sensors on the watch which record temperature in degrees Celsius. Note that the index/number for each corresponds to the index in the photo on the Overview tab.
- `tof_[1-5]_v[0-63]` - There are five time-of-flight sensors on the watch that measure distance. In the dataset, the 0th pixel for the first time-of-flight sensor can be found with column name `tof_1_v0`, whereas the final pixel in the grid can be found under column `tof_1_v63`. This data is collected row-wise, where the first pixel could be considered in the top-left of the grid, with the second to its right, ultimately wrapping so the final value is in the bottom right (see image above). The particular time-of-flight sensor is denoted by the number at the start of the column name (e.g., 1_v0 is the first pixel for the first time-of-flight sensor while 5_v0 is the first pixel for the fifth time-of-flight sensor). If there is no sensor response (e.g., if there is no nearby object causing a signal reflection), a -1 is present in this field. Units are uncalibrated sensor values in the range 0-254. Each sensor contains 64 pixels arranged in an 8×8 grid, visualized in the figure below.

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

■ 数据概览

[train/test]_demographics.csv

These tabular files contain demographic and physical characteristics of the participants.

- `subject`
- `adult_child` : Indicates whether the participant is a child (`0`) or an adult (`1`). Adults are defined as individuals aged 18 years or older.
- `age` : Participant's age in years at time of data collection.
- `sex` : Participants sex assigned at birth, `0` = female, `1` = male.
- `handedness` : Dominant hand used by the participant, `0` = left-handed, `1` = right-handed.
- `height_cm` : Height of the participant in centimeters.
- `shoulder_to_wrist_cm` : Distance from shoulder to wrist in centimeters.
- `elbow_to_wrist_cm` : Distance from elbow to wrist in centimeters.

sample_submission.csv

- `sequence_id`
- `gesture`

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

■ 评价指标

The evaluation metric for this contest is a version of macro F1 that equally weights two components:

1. **Binary F1** on whether the `gesture` is one of the target or non-target types.
2. **Macro F1** on `gesture`, where all non-target sequences are collapsed into a single `non_target` class

The final score is the average of the binary F1 and the macro F1 scores.

```
self.target_gestures = [
    'Above ear - pull hair',
    'Cheek - pinch skin',
    'Eyebrow - pull hair',
    'Eyelash - pull hair',
    'Forehead - pull hairline',
    'Forehead - scratch',
    'Neck - pinch skin',
    'Neck - scratch',
]
```

```
self.non_target_gestures = [
    'Write name on leg',
    'Wave hello',
    'Glasses on/off',
    'Text on phone',
    'Write name in air',
    'Feel around in tray and pull out an object',
    'Scratch knee/leg skin',
    'Pull air toward your face',
    'Drink from bottle/cup',
    'Pinch knee/leg skin'
]
```

Macro-F1（宏平均）

- **步骤**：
  1. 对**每个类别**单独计算F1值（即分别计算Precision和Recall，再调和平均）。
  2. 将所有类别的F1值**算术平均**，得到最终结果。
- **公式**：

$$\text{Macro-F1} = \frac{1}{N}\sum_{i=1}^{N}\text{F1}_i$$

（N为类别数，F1_i是第i类的F1值）

Micro-F1（微平均）

- **步骤**：
  1. **汇总所有类别的TP/FP/FN**（True Positive/False Positive/False Negative）。
  2. 用全局的TP、FP、FN计算**整体的Precision和Recall**，再求F1。
- **公式**：

$$\text{Micro-Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \sum \text{FP}}, \quad \text{Micro-Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \sum \text{FN}}$$

$$\text{Micro-F1} = 2 \cdot \frac{\text{Micro-P} \times \text{Micro-R}}{\text{Micro-P} + \text{Micro-R}}$$

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

■ **提交文件**

```
if not os.getenv('KAGGLE_IS_COMPETITION_RERUN'):
    print(pd.read_parquet("submission.parquet"))
```

```
  sequence_id              gesture
0  SEQ_000001  Eyebrow - pull hair
1  SEQ_000011  Eyelash - pull hair
```

Leaderboard

Raw Data   Refresh

Search leaderboard

Public   Private

This leaderboard is calculated with approximately 42% of the test data. The final results will be based on the other 58%, so the final standings may be different.

Prize Contenders

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

根据题目背景，这是一道典型的分类问题，具体通用思路如下：

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

LSTM模型 https://www.kaggle.com/code/jiazhuang/cmi-imu-only-lstm

特征工程 + NN模型

```
def feature_engineering(train_df):
    # IMU magnitude
    train_df['acc_mag'] = np.sqrt(train_df['acc_x']**2 + train_df['acc_y']**2 + train_df['acc_z']**2)

    # IMU angle
    train_df['rot_angle'] = 2 * np.arccos(train_df['rot_w'].clip(-1, 1))

    # IMU jerk, angular velocity
    train_df['acc_mag_jerk'] = train_df.groupby('sequence_id')['acc_mag'].diff().fillna(0)
    train_df['rot_angle_vel'] = train_df.groupby('sequence_id')['rot_angle'].diff().fillna(0)

    # Remove gravity
    def get_linear_accel(df):
        res = remove_gravity_from_acc(
            df[['acc_x', 'acc_y', 'acc_z']],
            df[['rot_x', 'rot_y', 'rot_z', 'rot_w']]
        )
        res = pd.DataFrame(res, columns=['linear_acc_x', 'linear_acc_y', 'linear_acc_z'], index=df.index)
        return res

    linear_accel_df = train_df.groupby('sequence_id').apply(get_linear_accel, include_groups=False)
    linear_accel_df = linear_accel_df.droplevel('sequence_id')
    train_df = train_df.join(linear_accel_df)
    train_df['linear_acc_mag'] = np.sqrt(train_df['linear_acc_x']**2 + train_df['linear_acc_y']**2 + train_df['linear_acc_z']**2)
    train_df['linear_acc_mag_jerk'] = train_df.groupby('sequence_id')['linear_acc_mag'].diff().fillna(0)

    # Calc angular velocity
    def calc_angular_velocity(df):
        res = calculate_angular_velocity_from_quat( df[['rot_x', 'rot_y', 'rot_z', 'rot_w']] )
        res = pd.DataFrame(res, columns=['angular_vel_x', 'angular_vel_y', 'angular_vel_z'], index=df.index)
        return res

    angular_velocity_df = train_df.groupby('sequence_id').apply(calc_angular_velocity, include_groups=False)
    angular_velocity_df = angular_velocity_df.droplevel('sequence_id')
    train_df = train_df.join(angular_velocity_df)

    # Calculating angular distance
    def calc_angular_distance(df):
        res = calculate_angular_distance(df[['rot_x', 'rot_y', 'rot_z', 'rot_w']])
        res = pd.DataFrame(res, columns=['angular_distance'], index=df.index)
        return res

    angular_distance_df = train_df.groupby('sequence_id').apply(calc_angular_distance, include_groups=False)
    angular_distance_df = angular_distance_df.droplevel('sequence_id')
    train_df = train_df.join(angular_distance_df)

    train_df[FEATURE_NAMES] = train_df[FEATURE_NAMES].ffill().bfill().fillna(0).values.astype('float32')

    return train_df
```

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

模型定义

```
class SEBlock(nn.Module):
    def __init__(self, channels, reduction=8):
        super().__init__()
        self.squeeze = nn.AdaptiveAvgPool1d(1)
        self.excitation = nn.Sequential(
            nn.Linear(channels, channels // reduction, bias=False),
            nn.ReLU(inplace=True),
            nn.Linear(channels // reduction, channels, bias=False),
            nn.Sigmoid()
        )

    def forward(self, x):
        b, c, _ = x.size()
        y = self.squeeze(x).view(b, c)
        y = self.excitation(y).view(b, c, 1)
        return x * y.expand_as(x)
```

```
class ResidualSECNNBlock(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size, pool_size=2, dropout=0.3, weight_decay=1e-4):
        super().__init__()

        # First conv block
        self.conv1 = nn.Conv1d(in_channels, out_channels, kernel_size, padding=kernel_size//2, bias=False)
        self.bn1 = nn.BatchNorm1d(out_channels)

        # Second conv block
        self.conv2 = nn.Conv1d(out_channels, out_channels, kernel_size, padding=kernel_size//2, bias=False)
        self.bn2 = nn.BatchNorm1d(out_channels)

        # SE block
        self.se = SEBlock(out_channels)

        # Shortcut connection
        self.shortcut = nn.Sequential()
        if in_channels != out_channels:
            self.shortcut = nn.Sequential(
                nn.Conv1d(in_channels, out_channels, 1, bias=False),
                nn.BatchNorm1d(out_channels)
            )

        self.pool = nn.MaxPool1d(pool_size)
        self.dropout = nn.Dropout(dropout)
```

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

模型定义

```python
def forward(self, x):
    shortcut = self.shortcut(x)

    # First conv
    out = F.relu(self.bn1(self.conv1(x)))
    # Second conv
    out = self.bn2(self.conv2(out))

    # SE block
    out = self.se(out)

    # Add shortcut
    out += shortcut
    out = F.relu(out)

    # Pool and dropout
    out = self.pool(out)
    out = self.dropout(out)

    return out
```

```python
class AttentionLayer(nn.Module):
    def __init__(self, hidden_dim):
        super().__init__()
        self.attention = nn.Linear(hidden_dim, 1)

    def forward(self, x):
        # x shape: (batch, seq_len, hidden_dim)
        scores = torch.tanh(self.attention(x))  # (batch, seq_len, 1)
        weights = F.softmax(scores.squeeze(-1), dim=1)  # (batch, seq_len)
        context = torch.sum(x * weights.unsqueeze(-1), dim=1)  # (batch, hidden_dim)
        return context
```

```python
class TwoBranchModel(nn.Module):
    def __init__(self, imu_dim, tof_dim, n_classes, weight_decay=1e-4):
        super().__init__()
        self.imu_dim = imu_dim
        self.tof_dim = tof_dim
        self.n_classes = n_classes
        self.weight_decay = weight_decay

        # IMU deep branch
        self.imu_block1 = ResidualSECNNBlock(imu_dim, 64, 3, dropout=0.3, weight_decay=weight_dec
ay)
        self.imu_block2 = ResidualSECNNBlock(64, 128, 5, dropout=0.3, weight_decay=weight_decay)
```

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

模型训练

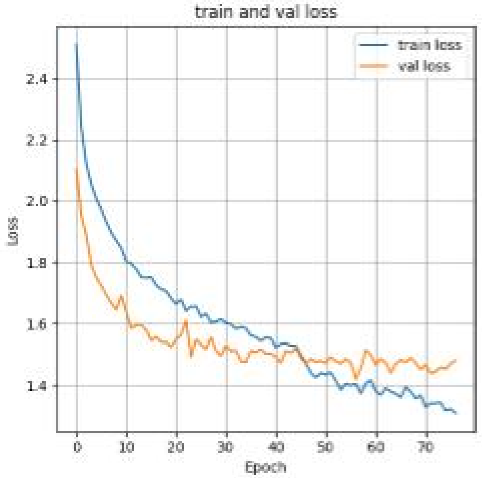
```
    def train(self, train_loader, val_loader, num_epochs=50, learning_rate=0.001,
              weight_decay=1e-5, patience=10, save_path='best_model.pth', mixup_augmenter=None):
        """完整训练流程"""

        # 优化器和损失函数
        optimizer = optim.Adam(self.model.parameters(), lr=learning_rate, weight_decay=weight_dec
ay)
        criterion = self.criterion

        # 学习率调度器
        scheduler = optim.lr_scheduler.ReduceLROnPlateau(
            optimizer, mode='max', factor=0.5, patience=patience//2
        )

        # 早停
        best_val_score = -float('inf')
        epochs_without_improvement = 0

        print(f"开始训练，设备：{self.device}")
        print(f"模型参数量：{sum(p.numel() for p in self.model.parameters()):,}")

        # 记录当前学习率用于检测变化
        current_lr = learning_rate
```

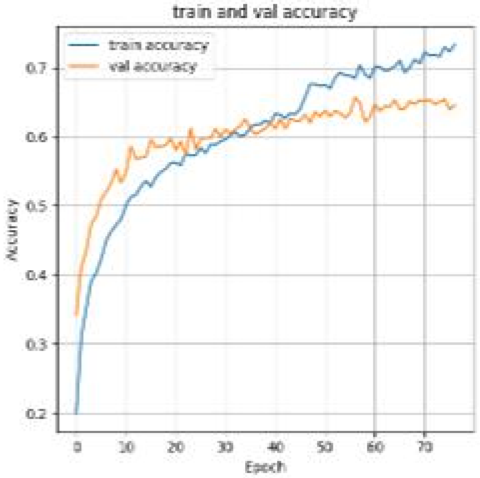
```
# 前向传播
optimizer.zero_grad()
outputs = self.model(sequences)

# 计算损失
if mixup_augmenter is not None and lambda_ != 1.0:
    loss = mixup_criterion(criterion, outputs, labels_a, labels_b, lambda_)
else:
    loss = criterion(outputs, labels)

# 反向传播
loss.backward()

# 梯度裁剪
torch.nn.utils.clip_grad_norm_(self.model.parameters(), max_norm=1.0)

optimizer.step()

# 统计
total_loss += loss.item()
_, predicted = torch.max(outputs.data, 1)
```

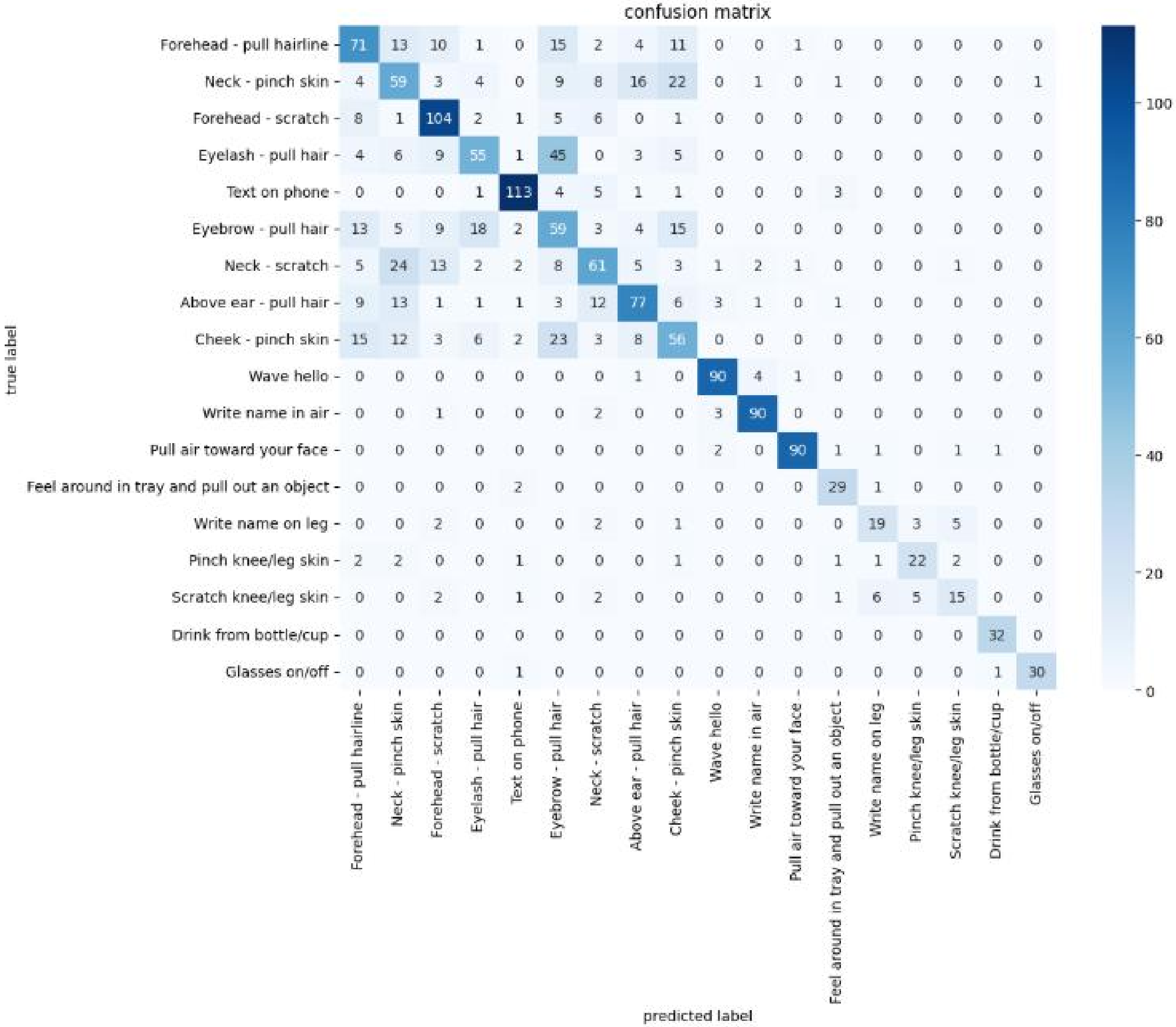
CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

模型结果







训练历史图保存到 ./saved_models/fold_2_training_history.png
测试准确率： 0.6569
测试分数： 0.7746

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

Bert+lstm/gru模型

https://www.kaggle.com/code/wasupandceacar/lb-0-841-5fold-single-model-with-split-sensors

```
        self.thm_branch1, self.tof_branch1 = self.init_thm_tof_branch(thm_dim//5, tof_dim//5, **kwargs)
        self.thm_branch2, self.tof_branch2 = self.init_thm_tof_branch(thm_dim//5, tof_dim//5, **kwargs)
        self.thm_branch3, self.tof_branch3 = self.init_thm_tof_branch(thm_dim//5, tof_dim//5, **kwargs)
        self.thm_branch4, self.tof_branch4 = self.init_thm_tof_branch(thm_dim//5, tof_dim//5, **kwargs)
        self.thm_branch5, self.tof_branch5 = self.init_thm_tof_branch(thm_dim//5, tof_dim//5, **kwargs)
```

```
    def feature_block(self, in_channels, out_channels, num_layers, pool_size=2, drop=0.3):
        return nn.Sequential(
            *[ResNetSEBlock(in_channels=in_channels, out_channels=in_channels) for i in range(num_layers)],
            nn.Conv1d(in_channels, out_channels, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm1d(out_channels),
            nn.ReLU(inplace=True),
            nn.MaxPool1d(pool_size, ceil_mode=True),
            nn.Dropout(drop)
        )
```

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

CNN+LSTM/GRU模型

https://www.kaggle.com/code/wasupandceacar/lb-0-841-5fold-single-model-with-split-sensors

```python
    def residual_feature_block(self, in_channels, out_channels, num_layers, pool_size=2, drop=0.3):
        return nn.Sequential(
            *[ResNetSEBlock(in_channels=in_channels, out_channels=in_channels) for i in range(num_layers)],
            ResNetSEBlock(in_channels, out_channels, wd=1e-4),
            nn.MaxPool1d(pool_size, ceil_mode=True),
            nn.Dropout(drop)
        )

    def init_thm_tof_branch(self, thm_dim, tof_dim, **kwargs):
        thm_branch = nn.Sequential(
            self.feature_block(thm_dim, kwargs["thm1_channels"], kwargs["thm1_layers"], drop=kwargs["thm1_dropout"]),
            self.feature_block(kwargs["thm1_channels"], kwargs["thm2_channels"], kwargs["thm2_layers"], drop=kwargs["thm2_dropout"]),
        )
        tof_branch = nn.Sequential(
            self.feature_block(tof_dim, kwargs["tof1_channels"], kwargs["tof1_layers"], drop=kwargs["tof1_dropout"]),
            self.feature_block(kwargs["tof1_channels"], kwargs["tof2_channels"], kwargs["tof2_layers"], drop=kwargs["tof2_dropout"]),
        )
        return thm_branch, tof_branch
```

CMI - Detect Behavior with Sensor Data

Predicting Body Focused Repetitive Behaviors from a Wrist-Worn Device

后处理

遍历寻找18个乘数

使得macroF1最大

```
Class 0: best multiplier = 0.95, F1 = 0.8592
Class 1: best multiplier = 1.0, F1 = 0.8592
Class 2: best multiplier = 0.75, F1 = 0.8593
Class 3: best multiplier = 1.05, F1 = 0.8593
Class 4: best multiplier = 0.85, F1 = 0.8595
Class 5: best multiplier = 1.15, F1 = 0.8596
Class 6: best multiplier = 1.05, F1 = 0.8598
Class 7: best multiplier = 1.2, F1 = 0.8600
Class 8: best multiplier = 0.8, F1 = 0.8601
Class 9: best multiplier = 1.0, F1 = 0.8601
Class 10: best multiplier = 1.0, F1 = 0.8601
Class 11: best multiplier = 1.05, F1 = 0.8602
Class 12: best multiplier = 0.95, F1 = 0.8602
Class 13: best multiplier = 1.15, F1 = 0.8603
Class 14: best multiplier = 0.9, F1 = 0.8604
Class 15: best multiplier = 0.85, F1 = 0.8605
Class 16: best multiplier = 0.9, F1 = 0.8607
Class 17: best multiplier = 0.75, F1 = 0.8608

array([0.95, 1.  , 0.75, 1.05, 0.85, 1.15, 1.05,
       1.05, 0.95, 1.15, 0.9 , 0.85, 0.9 , 0.75])
```

```python
def grid_search_multipliers(probabilities, true_labels, values_to_try=[0.5, 1.0, 2.0]):
    """
    简单的网格搜索（适用于快速测试）
    """
    n_classes = probabilities.shape[1]
    best_f1 = 0
    best_multipliers = np.ones(n_classes)

    # 为每个类别尝试不同的乘数
    for class_idx in range(n_classes):
        current_best_f1 = 0
        current_best_multiplier = 1.0

        for multiplier in values_to_try:
            test_multipliers = best_multipliers.copy()
            test_multipliers[class_idx] = multiplier

            adjusted_probs = probabilities * test_multipliers
            pred_labels = np.argmax(adjusted_probs, axis=1)
            f1, overall_binary_f1, overall_macro_f1 = competition_metric(true_labels, pred_labels)
            # f1 = f1_score(true_labels, pred_labels, average='macro', zero_division=0)

            if f1 > current_best_f1:
                current_best_f1 = f1
                current_best_multiplier = multiplier

        best_multipliers[class_idx] = current_best_multiplier
        best_f1 = current_best_f1
        print(f"Class {class_idx}: best multiplier = {current_best_multiplier}, F1 = {current_best_f1:.4f}")

    return best_multipliers, best_f1
```

答疑环节