

BrainBeats: 用于疲劳检测的便携式脑机接口设备

1. 作品简介

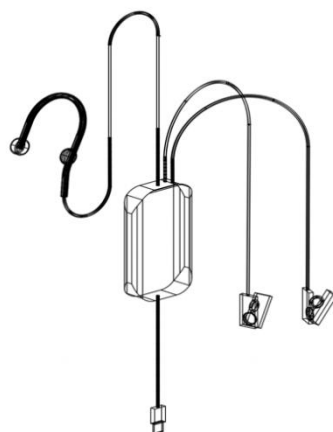


图 1 设备外观

该作品是一款用于疲劳检测的便携式脑机接口设备，该设备利用脑机接口技术，通过采集脑电信号、信号处理、外部信号控制等方式，对人体疲劳和清醒时的脑电波进行研究。设备仅需使用一个耳后单通道，就能够敏锐的捕捉脑电信号，并且提供用户友好的可视化界面，将大脑的疲劳状态展现在屏幕上，为用户提供了一种全新的，非侵入式的疲劳检测方式。

2. 设计背景及意义

2.1. 设计背景

在现代社会，人们的生活节奏越来越快，工作压力和学习负担逐渐增加，长时间的工作和学习可能导致身体和心理的疲劳。而疲劳状态下的驾驶和操作，不仅容易引发事故，还可能对个人的健康产生负面影响。因此，对个体的疲劳状态进行准确监测和及时干预，对于提高工作效率、确保交通安全和维护身体健康具有重要意义。

随着神经科学和生物医学工程的快速发展，脑电信号（EEG）作为一种反映大脑活动的生物电信号，被广泛应用于疲劳状态的监测。通过分析脑电信号的频谱和幅值等特征，可以

较为准确地判断个体的疲劳程度。传统的脑电监测设备通常较为庞大，不便携带，难以实现
在日常生活和工作中的实时监测。因此，一种便携式的基于脑电信号的疲劳监测设备势在必
行。

基于这一背景，我们设计了这款便携式疲劳监测设备。它采用了先进的脑电信号采集技
术，结合轻巧便携的硬件设计，旨在为用户提供一种方便、准确、可靠的疲劳监测解决方案。
通过该设备，个体用户可以随时随地监测自身的疲劳状态，以便在需要时及时调整工作和休
息，从而提高工作效率，确保安全驾驶，保护身体健康。

这款设备的设计不仅在技术层面具有创新性，还在用户体验和便携性方面进行了充分考
虑。它将科技与生活相结合，为人们的日常生活提供了一种智能化、便捷化的健康管理方式，
有望在提高生活质量和促进社会安全方面发挥积极作用。

2.2. 应用领域

这种便携式疲劳检测脑机接口设备拥有广泛的应用前景，其应用包括但不限于以下几个
方面：

1) 驾驶安全： 这种便携式设备可以在长时间驾车时监测驾驶者的疲劳状态。当设备检
测到驾驶者处于疲劳状态时，会发出警报，提醒驾驶者及时休息，从而降低因疲劳驾驶引发
的交通事故风险。

2) 工作环境： 在需要高度集中注意力的工作环境中，如操作重型机械、长时间计算或
处理复杂数据的工作，设备可以监测员工的疲劳状态，提醒他们适时休息，以防止意外事故
的发生。

3) 学习和考试： 学生在备考期间可能因长时间学习而疲劳。该设备可以帮助学生了解
自身疲劳程度，合理安排学习和休息时间，提高学习效率。

4) 日常生活： 适用于各种需要长时间保持警觉状态的活动，例如运动训练、瑜伽冥想

等。用户可以通过设备了解自身的疲劳程度，从而调整活动强度，确保身体和心理的健康。

5) 医疗领域： 在医疗领域，该设备可以用于监测患者的疲劳状态。特别是在一些康复训练中，疲劳监测可以帮助医生和患者掌握康复进程，避免过度劳累。

6) 心理健康： 对于一些焦虑症、抑郁症等心理健康问题的患者，该设备也可以用于监测他们的疲劳状况，从而合理安排休息和放松，帮助改善心理健康状态。

3. 设计方案

3.1. 硬件设计

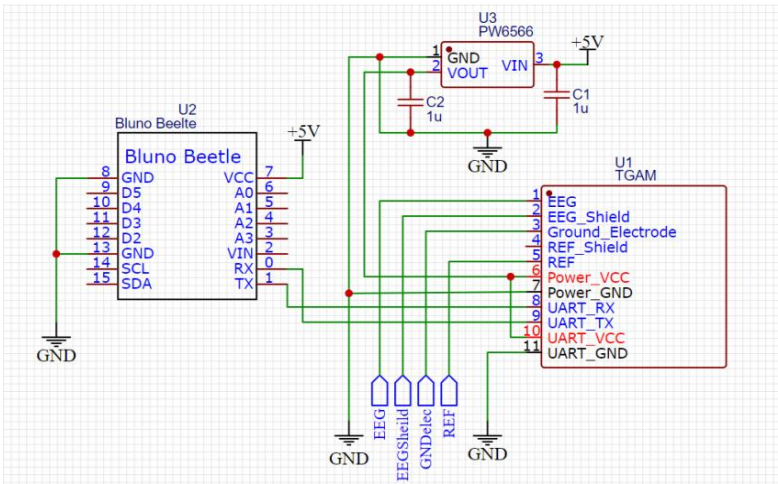
硬件部分需实现脑电信号采集、信号放大、滤波、模数转换和数据传输功能。脑电信号采集系统主要包含脑电信号采集、信号放大、滤波、模数转换功能。由于脑电信号是一种幅值很小的非平稳性信号，因此脑机接口设备对于放大电路有很高的需求，前置放大电路须保证直流偏移低、开路增益高、共模抑制比高以及阻抗高，适用于脑电采集的模数转换器通常要具有较低的功耗、较高的分辨率以及一定的噪声调制功能。为采集高质量脑电信号，选取集成了前置放大电路、滤波器、模数转换器及数据接口的 TGAM 脑电信号采集模块作为本设备的采集模块，该模块可直接输出原始脑电信号，且信号质量高、体积小、成本低廉，适用于单通道日用脑机接口设备的设计开发。同时，TGAM 可直接通过 UART 与控制器通讯，操作简单，信号传输稳定；我们采用已有的产品级开发板，以轻巧便捷、佩戴舒适、数据测量准确为目标，设计了可实现全套功能的第一套便携式脑电信号采集设备：

1) 采样电极：非侵入式干电极；TGAM 脑电信号采集模块可直接连接干电极并进行信号采样，因此，选用非侵入式干电极作为采样电极，干电极具有良好的导电效果，有效防止与皮肤接触过程中汗液皮脂油渍等污染电极导致导电率下降测量不准确的问题，同时，使用简单，佩戴方便，适用于便携式脑机接口设备。

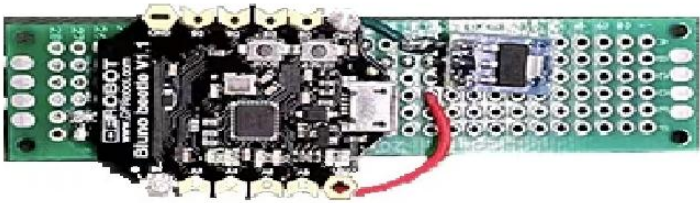
2) 脑电信号采集模块: neurosky 公司的 TGAM 模块。该模块集成前置放大电路、滤波器及模数转换器,并具有低功耗、分辨率高及噪声调制功能,且成本低廉,适用于单通道日用脑机接口设备开发;

3) 微控制器: DFRobot 公司产品 Bluno Beetle 开发板。Bluno Beetle 开发板是目前最小的同时集成了蓝牙 4.0 和 ATmega328P 的开发板,适用于可穿戴设备的设计开发,在 Bluno Beetle 开发板中,接收来自 TGAM 的脑电信号数据包,将数据包解码为原始脑电数据后,通过蓝牙 4.0 模块直接将原始数据传输至上位机。

第一版设备连接图如下所示: 在 Bluno Beetle 开发板中,接收来自 TGAM 的脑电信号数据包,将数据包解码为原始脑电数据后,通过蓝牙 4.0 模块直接将原始数据传输至上位机。



电路连接图

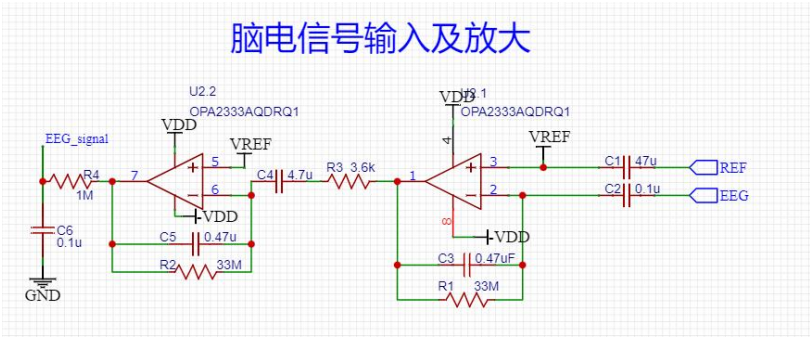


产品实物图

由于第一版硬件设备使用现有开发板，各部件无法继续缩小，不同部件间使用外部连线，连线松动易使信号传输不稳定，且未完全实现数据无线传输；因此为提升硬件性能，在确保数据准确性和传输稳定性的同时增加设备便携性，计划将整套电路集成在一块 PCB 板上：

1) 为保障佩戴舒适性，仍选用导电效果良好的非侵入式干电极作为采样电极；

2) 电路设计采用 TGAM 脑电采集模块和 Bluno Beetle 开发板进行。Bluno Beetle 开发板直接使用电源供电，通过稳压器 PW6566 将 5V 的供电电压降至 3.3V 作为 TGAM 脑电采集模块的供电电压。Bluno Beetle 开发板的 Tx 与 Rx 端口分别与 TGAM 脑电采集模块的 Rx 与 Tx 端口相连，TGAM 脑电采集模块的 Ground_Electrode 端口连接接地电极，EEG_Sheild 端口连接干电极外的保护层，REF 端口连接参考电极，EEG 端口直接连接干电极。采用 OPA2333 芯片搭建两极信号放大器，使脑电信号实现约 60dB 增益。该芯片功耗低，满足长时间应用需求；精度高、温度稳定性好、噪声低，能提供精确的信号放大处理，可提供清晰的信号放大，减小信号中的干扰，可满足对于微弱脑电信号放大需求。同时其具有低输入偏置电流，可减小对信号源的干扰，提高信号处理精度；



两极放大器示例

3) MCU 部分：nRF24LU1 芯片。nRF24LU1 是一款集成了 2.4GHz 无线通信和微控制器功能的单芯片解决方案，简化了整套电路设计，且该芯片高度集成，内置丰富的外设，可满足 AD 转换功能及无线传输功能，减小 PCB 板尺寸；且芯片采用低功耗设计，适用于电池供电应用，可延长设备使用时间；同时其具备 2.4GHz 无线通信模块，支持良好抗干扰性无线连

接，满足无线传输需求。

4) nRF24L01 接收器，市面上易于得到，价格低廉，传输速率高。



Nrf24l01 无线接收器

3) 控制器选取

微控制器在本设备中承担将采集的脑电信号传输至上位机的功能。本产品选用 DFEobot 生产的 Bluno Beetle 开发板为控制器，Bluno Beetle 开发板是目前最小的同时集成了蓝牙 4.0 和 ATmega328P 的开发板，适用于可穿戴设备的设计开发。可进行蓝牙数据传输，支持 USB 及无线编程，同时带有 V 型 I/O 接口，便于固定，使用便捷。在 Bluno Beetle 开发板中，接收来自 TGAM 的脑电信号数据包，将数据包解码为原始脑电数据后，通过蓝牙 4.0 模块直接将原始数据传输至上位机。

3.2. 数据集可行性

本项目数据已经全部完成采集。具体采集实验如下：

实验准备工作

脑电信号电平非常微弱，且容易受到环境噪声、生理活动噪声污染，因此在采集过程中要尽量控制噪声源，提升采集到的信号质量。在本次采集实验中，对噪声控制方法主要有以下几点：

- 1、所有采集使用相同的外部环境和设备仪器，选择安静且外部干扰少的时间和地点进行采集。
- 2、通过与被试沟通，减少其实验中容易产生噪声的行为，例如说话和其他幅度较大的动作。

3、合理设计实验流程和次数，避免采集过程中被试出现紧张、烦躁等状态。

采集设备的选取

为了测试本项目研发设备的采集质量，采集设备选用 BrainBeats，该设备支持单通道的采集实验，选用 T7/T8 通道采集。本设备采用干电极，避免了传统湿电极需要涂抹导电凝胶的步骤，极大提高了实验的效率。

数据采集流程

本次实验一共有 20 名被试，男性 10 名，女性 10 名。被试者均身体状态良好，无神经系统疾病病史。实验保证公开透明，在每次采集实验前，每位实验对象需认真阅读学习实验流程和实验注意事项，并签署知情同意书。实验在被试疲劳与清醒状态下分别采集被试的脑电数据。具体采集流程为：首先确定被试的 T7 通道位置，并用酒精擦拭，再用医用胶带贴上电极并固定，将参考电极耳夹夹在被试耳垂位置。调试设备直至脑电采集前端界面上的脑电数据稳定。在实验过程中保持被试头部稳定，双眼注视屏幕，播放一个时长约为 12 分钟的视频，并要求被试遵从视频指示，先集中注意 4 分钟，再闭眼休息 4 分钟，再集中注意 4 分钟。采集后需负责实验人员确认本次数据是否合格，如脑电数据波动起伏较大或电极有明显脱落，需要重新实验。此实验总时间约为 30 分钟。

脑电信号的预处理：

目前脑电信号（EEG）被广泛应用在各项研究中，但脑电信号通常非常微弱，并且在采集时难免混入其他噪声（如眼动信号、肌电信号等）。此外，采集设备和采集环境中的电磁噪声也会影响信号的质量。这就导致了脑电信号具有噪声多、信噪比低等特点。在本实验中，需

要参考现有信号处理方法，结合信号采集的实际情况寻找适合的信号处理流程，抑制信号中的噪声，提高信噪比，为特征提取工作做准备。

1. 工频降噪

工频降噪是一种常用的信号处理技术，旨在消除信号中的工频干扰噪声，即 50Hz 的周期性信号。这种噪声通常来自于电力线路中的电磁辐射，在信号采集和传输中，工频信号会混入噪声中，导致信号质量下降。因此需要通过陷波滤波对这些频段的噪声进行抑制。

2. 汉明窗与频域分析

由于一次性采集的时间较长，数据较多，需要将数据切片处理。普通的矩形窗口容易导致频谱泄漏，因此选择通过汉明窗进行数据的切片处理。汉明窗是一种常用的窗函数，可以有效减少频谱泄漏和抑制频谱副瓣。首先将信号数据与汉明窗进行逐元素的乘法操作，将数据加窗；然后对加窗后的信号数据进行快速傅里叶变换，得到频域表示与频率轴，方便进行后续的频谱分析和处理。

3. 滤波

在人类的脑电信号中，按频率划分可以分为以下不同频段：

频段	位置	频带介绍
δ 频段 (0.5-3Hz)	额部	常在睡眠时出现。
θ 频段 (4-7Hz)	主要在额区、中央区、颞区， 顶区有少量	在睡眠时增强，与思维活跃及专注程度相关。
α 频段 (8-13Hz)	主要在枕区	正常人静止或闭眼时较为明显，睁眼或接受其他刺激

		时消失。
β 频段 (14-30Hz)	主要在中央区和额区	在中枢神经系统强烈活动 或精神紧张、心情激动亢奋 时出现。
γ 频段 (>30Hz)	主要在额区及中央	由注意或感觉刺激引起，在 高度激动状态时出现。

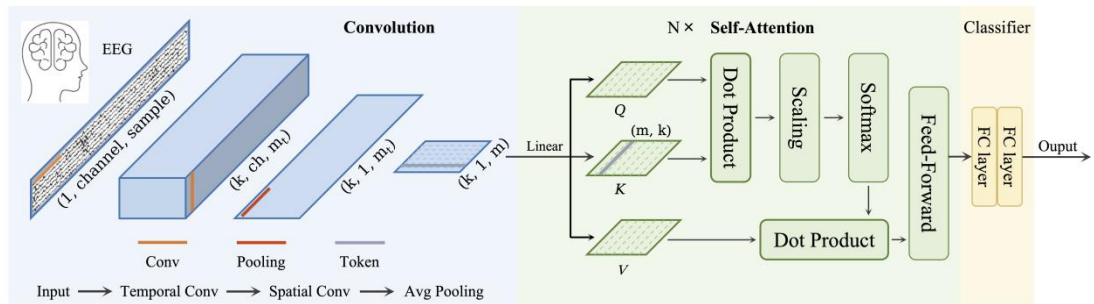
本实验使用以上全部频段，因此在 0.5-45Hz 频带范围内使用巴特沃斯(Butterworth)滤波器对脑电信号进行提取。巴特沃斯滤波器是一种常用的无限冲激响应(IIR)滤波器，具有平坦的频率相应和良好的滤波特性。通过滤波，可以突出特定范围内所要研究的脑电活动。

3.3. 分类模型

脑机接口（BCI）通过将神经活动作为控制信号，实现与计算机的直接通信。这种神经信号通常从多种经过深入研究的脑电图（EEG）信号中选择。对于给定的 BCI 范式，特征提取器和分类器根据预期的 EEG 控制信号的独特特征进行定制，限制了其应用于特定信号。卷积神经网络（CNN）已经被用于计算机视觉和语音识别，以实现自动特征提取和分类，并成功应用于基于 EEG 的 BCI。在本项目中，我们使用一个单一的 CNN 架构，准确地对 BCI 范式的 EEG 信号进行分类，并同时使其尽可能紧凑（模型中的参数数量尽量少）。

在这项工作中，我们使用了 Conformer，一种用于基于 Transformer 的 EEG 的 BCI 分类和解释神经网络。同时我们引入了在计算机视觉中使用的深度可分离卷积，用于构建一个特定于 EEG 的网络。

Conformer 的结构就是先通过一个卷积模块，提取局部特征，这一步可以适合大部分的 BCI 范式，同时对于类似于脑电波谱图等图像信息也有较强的接受并提取特征的能力。接着是一个自注意力机制模块，这个模块的主要功能就是提取 EEG 信号的全局特征，并关注 EEG 信号时序的全局相关性，更好的实现分类或者回归任务。



总体框架如图所示。该体系结构由三个组件组成：卷积模块、自注意模块和全连接分类器（本实验中我将其修改为回归模块以适应特定的任务）。在卷积模块中，以原始的二维脑电数据试验作为输入，分别沿着时间维度和电极通道维度应用时间卷积层和空间卷积层。然后，利用平均池化层在抑制噪声干扰的同时提高泛化能力。其次，将卷积模块得到的时空表示输入自注意模块。自我注意模块通过测量特征图中不同时间位置之间的全局相关性，进一步提取了长期的时间特征。最后，采用一个由多个全连接层组成的紧凑回归模块来输出解码结果。

下面具体介绍实现方法：

卷积模块

可以在 forward 函数中看到前向计算的方式：

在前向传播过程中，输入张量 x 首先经过一个维度调整操作，然后将其转换为浮点型。接着，将输入张量传入 shallownet 层序列和 projection 层序列进行处理，最后返回处理后的张量。

shallownet 属性是一个由多个层组成的序列，用于将输入张量进行浅层处理。具体而言，它包括了两个二维卷积层 (nn.Conv2d)、一个批归一化层 (nn.BatchNorm2d)、一个 ELU 激活函数 (nn.ELU)、一个平均池化层 (nn.AvgPool2d) 和一个 Dropout 层 (nn.Dropout)。这些层的作用是对输入张量进行特征提取和降维。

projection 属性是一个由多个层组成的序列，用于将处理后的特征进行投影。它包括一个二维卷积层 (nn.Conv2d) 和一个 Rearrange 操作。二维卷积层用于对特征进行变换，而 Rearrange 操作用于调整张量的维度顺序。

```

1. class PatchEmbedding(nn.Module):
2.     def __init__(self, emb_size=40):
3.         super().__init__()
4.
5.         self.shallownet = nn.Sequential(
6.             nn.Conv2d(17, 40, (25, 1), (1, 1)),
7.             nn.Conv2d(40, 40, (1, 1), (1, 1)),
8.             nn.BatchNorm2d(40),
9.             nn.ELU(),
10.            nn.AvgPool2d((1, 1), 1), # pooling acts as slicing to
            obtain 'patch' along the time dimension as in ViT
11.            nn.Dropout(0.5),
12.        )
13.
14.        self.projection = nn.Sequential(
15.            nn.Conv2d(40, emb_size, (1, 1), stride=(1, 1)), # tra
            nspose, conv could enhance fitting ability slightly
16.            Rearrange('b e (h) (w) -> b (h w) e'),
17.        )
18.
19.    def forward(self, x: Tensor) -> Tensor:
20.        b, _, _, _ = x.shape
21.        # x = x.squeeze(dim=-1) # 调整输入维度
22.        x = x.float()
23.        x = self.shallownet(x)
24.        x = self.projection(x)
25.        return x

```

总的来说，PatchEmbedding 类用于将输入张量进行特征提取和降维，并通过投影操作调整特征张量的维度顺序。这是视觉 transformer 中常用的操作之一。

Transformer 模块

此模块由多个 TransformerEncoderBlock 组成代码如下，其中 depth 代表模块数量，而 emb_size 代表输入和输出的特征维度。

```

1. class TransformerEncoder(nn.Sequential):
2.     def __init__(self, depth, emb_size):
3.         super().__init__([TransformerEncoderBlock(emb_size) for _
            in range(depth)])

```

接着介绍 TransformerEncoderBlock 的实现方法，

```

1. class TransformerEncoderBlock(nn.Sequential):

```

```

2.     def __init__(self,
3.                   emb_size,
4.                   num_heads=10,
5.                   drop_p=0.5,
6.                   forward_expansion=4,
7.                   forward_drop_p=0.5):
8.         super().__init__(
9.             ResidualAdd(nn.Sequential(
10.                nn.LayerNorm(emb_size),
11.                MultiHeadAttention(emb_size, num_heads, drop_p),
12.                nn.Dropout(drop_p)
13.            )),
14.            ResidualAdd(nn.Sequential(
15.                nn.LayerNorm(emb_size),
16.                FeedForwardBlock(
17.                    emb_size, expansion=forward_expansion, drop_p=
18.                    forward_drop_p),
19.                nn.Dropout(drop_p)
20.            ))

```

`__init__` 函数初始化 `TransformerEncoderBlock` 类，并接受多个参数：`emb_size`、`num_heads`、`drop_p`、`forward_expansion` 和 `forward_drop_p`。`emb_size` 表示输入和输出的特征维度，`num_heads` 表示多头注意力机制中的头数，`drop_p` 表示应用在注意力机制和前馈网络中的 dropout 概率，`forward_expansion` 表示前馈网络中的扩展因子，`forward_drop_p` 表示前馈网络中的 dropout 概率。

`super().__init__` 调用了父类 `nn.Sequential` 的构造函数，初始化了 `TransformerEncoderBlock` 类的实例，并接受两个子模块作为参数。

第一个子模块是一个 `ResidualAdd` 模块，包含了一个由多个层组成的序列。这个序列包括层归一化 (`nn.LayerNorm`)、多头注意力机制 (`MultiHeadAttention`) 和 dropout 层 (`nn.Dropout`)。这些层按顺序组合在一起，并通过 `nn.Sequential` 封装起来。

第二个子模块也是一个 `ResidualAdd` 模块，包含了一个由多个层组成的序列。这个序列包括层归一化 (`nn.LayerNorm`)、前馈网络模块 (`FeedForwardBlock`) 和 dropout 层 (`nn.Dropout`)。这些层按顺序组合在一起，并通过 `nn.Sequential` 封装起来。

总的来说，`TransformerEncoderBlock` 类是一个完整的 Transformer 编码块，由多个层组成。它包括了一个多头注意力机制和一个前馈网络，并使用残差连接和层归一化来优化

信息传递和梯度流动。

这里使用了多头注意力机制，其实就是多个自注意力机制的结合，下面具体解释如何实现多头注意力机制：

```
1. class MultiHeadAttention(nn.Module):
2.     def __init__(self, emb_size, num_heads, dropout):
3.         super().__init__()
4.         self.emb_size = emb_size
5.         self.num_heads = num_heads
6.         self.keys = nn.Linear(emb_size, emb_size)
7.         self.queries = nn.Linear(emb_size, emb_size)
8.         self.values = nn.Linear(emb_size, emb_size)
9.         self.att_drop = nn.Dropout(dropout)
10.        self.projection = nn.Linear(emb_size, emb_size)
11.
12.    def forward(self, x: Tensor, mask: Tensor = None) -> Tensor:
13.        # x = np.expand_dims(x, axis=3)
14.        queries = rearrange(self.queries(x), "b n (h d) -> b h n d", h=self.num_heads)
15.        keys = rearrange(self.keys(x), "b n (h d) -> b h n d", h=self.num_heads)
16.        values = rearrange(self.values(x), "b n (h d) -> b h n d", h=self.num_heads)
17.        energy = torch.einsum('bhqd, bhkd -> bhqk', queries, keys)
18.
19.        if mask is not None:
20.            fill_value = torch.finfo(torch.float32).min
21.            energy.mask_fill(~mask, fill_value)
22.
23.        scaling = self.emb_size ** (1 / 2)
24.        att = F.softmax(energy / scaling, dim=-1)
25.        att = self.att_drop(att)
26.        out = torch.einsum('bhal, bhlv -> bhav ', att, values)
27.        out = rearrange(out, "b h n d -> b n (h d)")
28.        out = self.projection(out)
29.        return out
```

`__init__` 函数初始化 `MultiHeadAttention` 类,并接受三个参数:`emb_size`、`num_heads` 和 `dropout`。`emb_size` 表示输入和输出的特征维度, `num_heads` 表示多头注意力机制中的头数, `dropout` 表示应用在注意力权重上的 `dropout` 概率。在 `__init__` 函数中,通过 `nn.Linear` 创建了三个线性层 `keys`、`queries` 和 `values`, 用于将输入特征进行线性变换。这些线性层的输入和输出维度都是 `emb_size`。`att_drop` 是一个 `dropout` 层, 应用在注意

力权重上。`projection` 是一个线性层，用于将多头注意力的输出进行线性变换。`forward` 函数定义了多头注意力机制的前向传播过程。在前向传播过程中，输入张量 `x` 经过线性变换 `queries`、`keys` 和 `values`，并通过 `rearrange` 函数调整张量的维度顺序，以便进行多头注意力计算。通过 `torch.einsum` 函数，计算注意力权重 `att`，并根据需要应用掩码 `mask`。然后，将注意力权重与值进行加权求和，并通过 `rearrange` 和线性变换 `projection` 进行输出的调整和变换。

回归模块

最后是回归模块，论文中本来这里是分类模块，但是由于使用的回归数据集在具体的新的任务上进行测试，因此这里需要修改相应的代码：

```
1. class RegressionHead(nn.Sequential):
2.     def __init__(self, emb_size):
3.         super().__init__()
4.         self.clshead = nn.Sequential(
5.             nn.AdaptiveAvgPool1d(1), # 全局平均池化
6.             nn.Flatten(),
7.             nn.LayerNorm(emb_size),
8.             nn.Linear(emb_size, 1)
9.         )
10.        self.fc = nn.Sequential(
11.            nn.Linear(40, 256),
12.            nn.ELU(),
13.            nn.Dropout(0.5),
14.            nn.Linear(256, 32),
15.            nn.ELU(),
16.            nn.Dropout(0.3),
17.            nn.Linear(32, 1)
18.        )
19.        def forward(self, x):
20.            x = x.contiguous().view(x.size(0), -1)
21.            out = self.fc(x)
22.            out = torch.sigmoid(out)
23.            return x, out
```

首先，`__init__` 函数初始化 `RegressionHead` 类，并接受一个参数 `emb_size`，表示输入特征的维度。在 `__init__` 函数中，首先定义了 `clshead` 子模块，它由一系列层组成。这些层包括自适应平均池化层 (`nn.AdaptiveAvgPool1d`)、展平层 (`nn.Flatten`)、层归一化层 (`nn.LayerNorm`) 和线性层 (`nn.Linear`)。这些层的作用是将输入特征进行降维和映射，

最终输出一个标量值。接下来，定义了 `fc` 子模块，也是由一系列层组成。这些层包括线性层、ELU 激活函数层 (`nn.ELU`)、dropout 层 (`nn.Dropout`)。这些层的作用是对输入特征进行一系列线性变换和非线性激活，以生成最终的回归输出。`forward` 函数定义了前向传播过程。在前向传播过程中，输入张量 `x` 首先通过 `contiguous` 函数进行连续化操作，然后通过 `view` 函数将其形状调整为 `(batch_size, -1)`，以适应后续的线性层输入要求。接着，将调整后的张量输入到 `fc` 子模块中进行计算，并通过 `torch.sigmoid` 将输出映射到 `(0, 1)` 范围内（这是收到数据集的标签的影响，`perclos` 的范围是在 0-1 之间），以得到回归的预测输出。

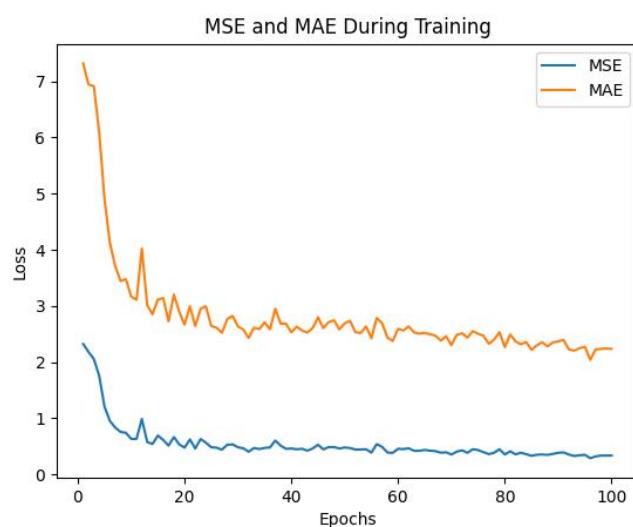
实验结果验证

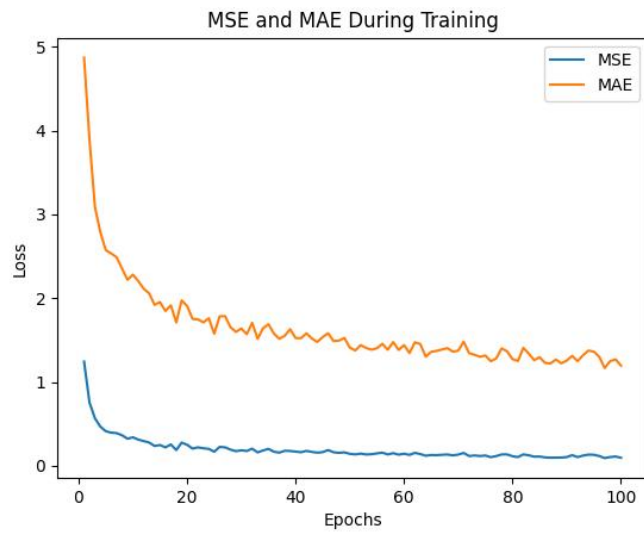
测试 23 个被试的实验数据，epoch 设置为 100，batch_size 设置为 32，测试 23 个被试数据并计算验证集的平均 MSE，MAE 和方差，以此来衡量模型的效果（具体的代码可见压缩包中的 main.py 这里就不赘述展开）：

被试编号	MSE	MAE	MSE_Var	MAE_Var
1_20151124_noon_2	0.3142	0.4844	0.0385	0.0800
2_20151106_noon	0.0064	0.0608	0.0002	0.0027
3_20151024_noon	0.0041	0.0504	2.6038e-05	0.0015
4_20151105_noon	0.0298	0.1411	0.0013	0.0100
4_20151107_noon	0.0441	0.1593	0.0036	0.0188
5_20141108_noon	0.0823	0.1761	0.0301	0.0516
5_20151012_night	0.0156	0.0928	0.0019	0.0070
6_20151121_noon	0.0122	0.0867	0.0002	0.0048
7_20151015_night	0.0048	0.0577	2.7783e-05	0.0015
8_20151022_noon	0.2661	0.4240	0.0517	0.0868
9_20151017_night	0.0149	0.0740	0.0013	0.0094
10_20151125_noon	0.0199	0.1211	0.0005	0.0053
11_20151024_night	0.0105	0.0790	0.0003	0.0043
12_20150928_noon	0.0393	0.0941	0.0189	0.0306
13_20150929_noon	0.0385	0.1699	0.0021	0.0097
14_20151014_night	0.0398	0.1399	0.0061	0.0203
15_20151126_night	0.0055	0.0534	9.0361e-05	0.0027
16_20151128_night	0.0495	0.0979	0.0295	0.0401
17_20150925_noon	0.0057	0.0575	8.5449e-05	0.0024
18_20150926_noon	0.0205	0.0744	0.0043	0.0151
19_20151114_noon	0.0070	0.0666	7.8169e-05	0.0026
20_20151129_night	0.0007	0.0197	1.2887e-06	0.0003
21_20151016_noon	0.1975	0.3076	0.0923	0.1034
average	0.0534	0.1343	0.0058	0.0222

从以上结果可以看出，本模型展现出了可靠的拟合能力，并成功实现了对给定脑电范式的预测。同时其也表现出来了惊人的快速拟合能力，大约在 20 个轮次的训练之后模型的 MSE 就已经足够低，这一点表现远超 EEGNet，后者大约需要 1200 甚至 2000 个轮次才能将损失降下来，可见加入了 Transformer 之后对于不同的 BCI 范式的泛化能力大大的增强了，或许这从某种角度来说也可以证明脑电信号前后关联性对于脑电分类任务有极其重大的意义，人脑的思维在某种程度上是会受到前置事件的影响的。同时将此模型训练过程中的 MSE 和 MAE 进行可视化，我们可以发现并不存在任何像之前 EEGNet 中提到的关于损失值忽然波动的问题，可见此模型的拟合能力远强于 EEGNet，也没有出现 EEGNet 中个别被试无法成功收敛损失值的情况。综上，Conformer 表现出色且准确率高，或许是未来主流的对于脑电进行处理的手段之一。

下面展示本次训练中的部分被试的 mse 和 mae 的训练和验证集拟合曲线(全部图片可见压缩包)：





3.4. 前端结果输出交互设计

1) 登录界面设计

在用户进入程序时默认的用于账号登录界面。用户可以通过该界面进行登录账号的操作。

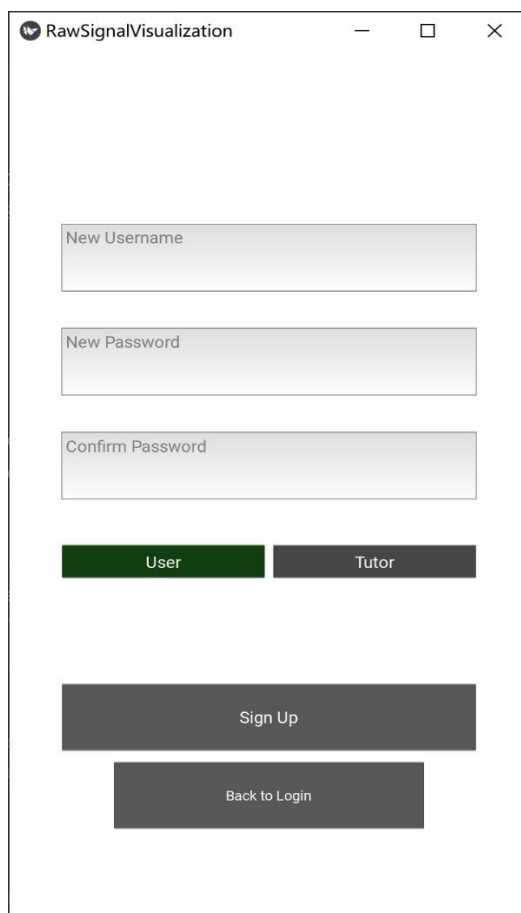
产品的 logo 也在登录界面上体现。当输入的用户名与密码都正确时，用户才被准许成功登录。

The screenshot shows a web application window titled 'RawSignalVisualization'. The interface includes the 'BrainBeats' logo at the top. Below the logo are two input fields labeled 'Username' and 'Password'. At the bottom, there are two buttons: 'Login' and 'Sign Up'.

图 3 登录界面

2) 注册界面设计

该便携式疲劳检测脑机接口设备对于每位初次使用该设备的用户都要求通过该注册界面来新建账号，以存储该用户的所有脑电数据信息。通过注册界面，用户可以设定用户名、密码，并确认密码。新注册的用户信息将被储存。



The image shows a registration window titled "RawSignalVisualization". It contains three input fields for "New Username", "New Password", and "Confirm Password". Below these fields are two buttons: "User" (highlighted in green) and "Tutor" (dark grey). At the bottom, there are two more buttons: "Sign Up" and "Back to Login".

图 4 注册界面

3) 功能选择界面设计

通过该界面可以选择当前需要的功能。用户可以选择开始记录脑电数据、回看脑电数据和退出当前登录的账号。

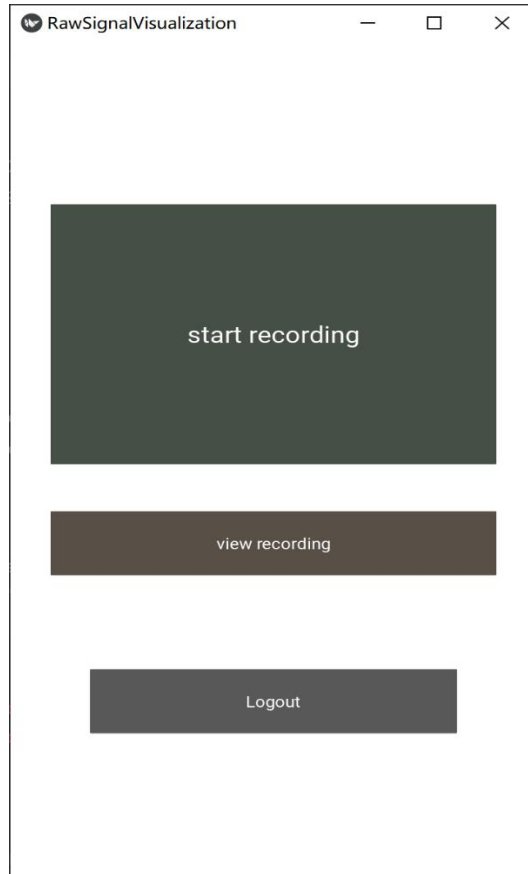


图 5 功能选择界面

4) 实时脑电波形显示界面设计

通过读取上位机连接硬件的对应端口传输进来的数据，绘制波形图。将上位机与硬件用数据线相连接之后，数据线会将脑电原始数据实时绘制成波形图，以展现给用户。横轴为时间（以秒为单位），纵轴为脑电原始数据。在波形图下方，程序会根据实时的分类结果判断当前为疲劳状态还是清醒状态。所记录的导电数据还会写入一个以开始记录的时间命名的 csv 文件中，方便后续回看。

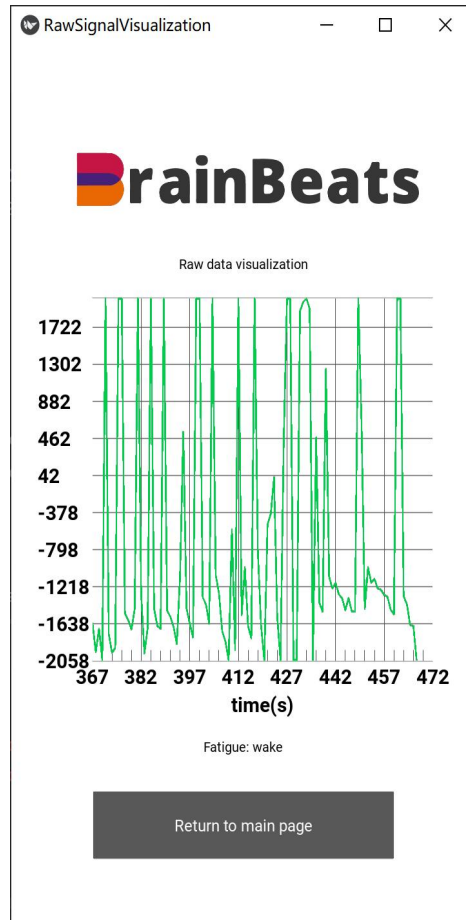


图 6 实时脑电波形显示界面

3.4. 设备外观设计

本设备外观设计遵循轻便易携带、简单易佩戴的中心思路，在保证设备功能性的基础上将整体重量与体积降到最低，以完善用户体验。

设备外壳整体结构主要由四部分组成，具体指图 9 中 1：耳挂与脑电信号采集电极，2：采集模块，3：电源及数据传输接口，4 与 5：左右耳参考电极。

其中，采集模块由外壳覆盖以保护和固定电路结构，增强设备可携带性。该部分具体指图 10 中 6：电路板，7：电源及数据传输线，8：降压芯片，9：usb 接口，10：Bluno Beetle 开发板，11：脑电信号采集线，12：TGAM 脑电信号采集模块，13：外壳，14：支撑结构。

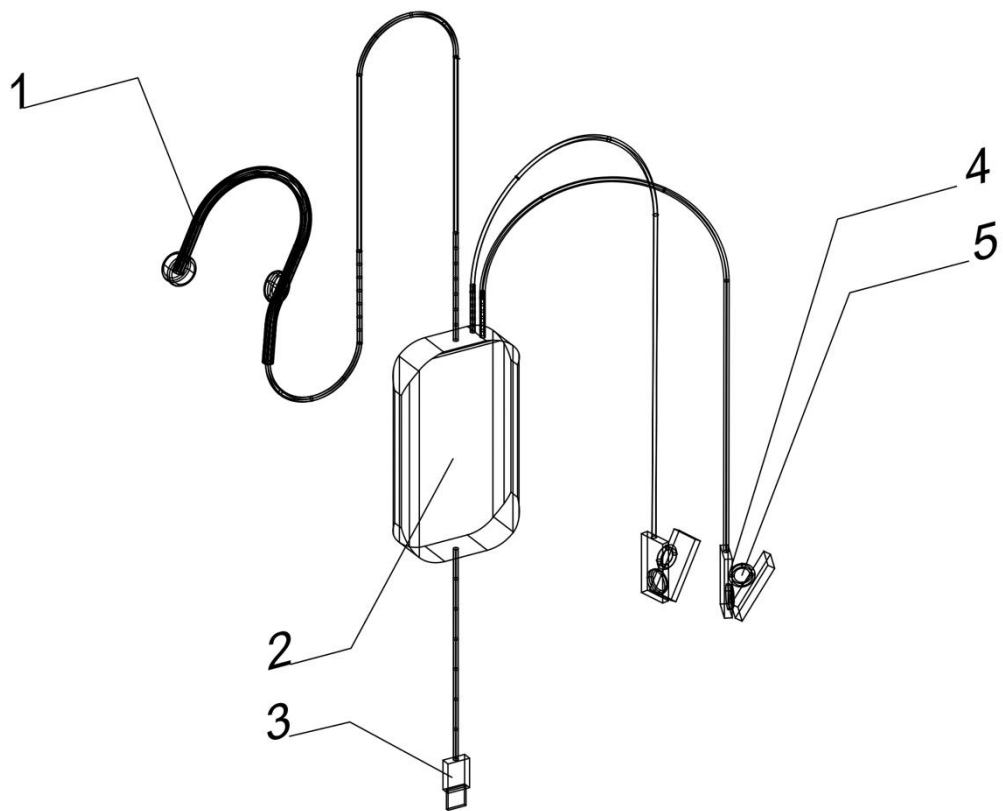


图 7 设备结构示意图

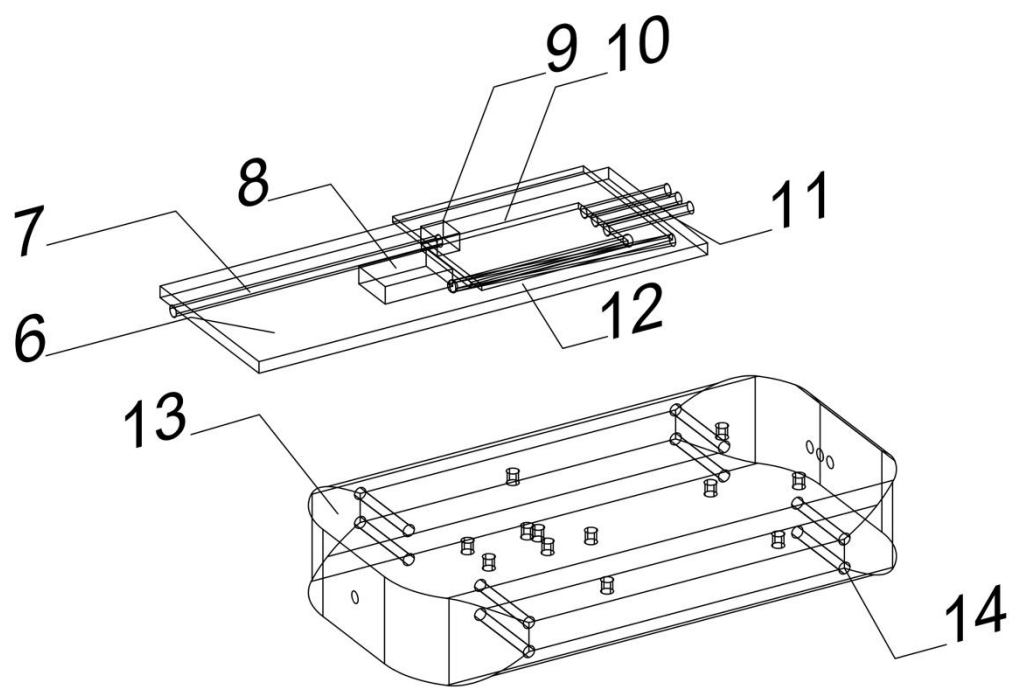


图 8 部分立体结构示意图

4. 工作原理

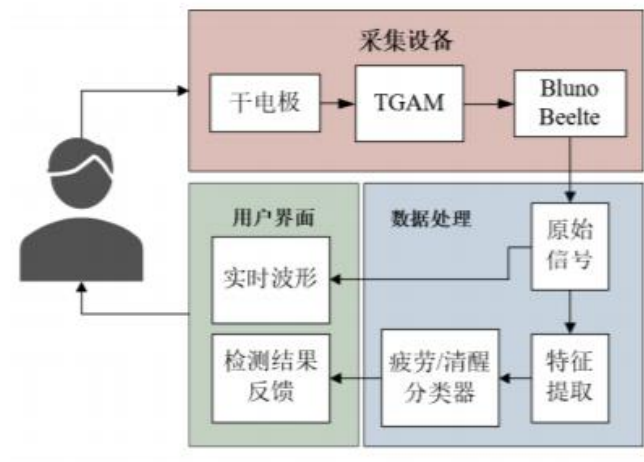


图 9 工作流程图

4.1. 脑电信号采集及信号初步处理

该设备采用干电极作为脑电传感器，使用设备采集脑电信号时，电极贴在用户耳后 L3 采集电位的皮肤上，干电极可以感知到大脑活动时产生的微弱神经信号，即脑电信号（EEG 信号），这些信号通常是由大脑神经元的电活动引起的，并且反映了不同脑区的活跃度。

脑电信号采集后，会被传输至设备中的 TGAM 脑电信号采集模块，在该模块中，脑电信号会经过放大、滤波和去噪等处理步骤。这些处理步骤旨在提供出用户特定的脑电模式，去除噪声，从而得到清晰、可靠的信号。

TGAM 所采集处理后的数据传输至 Bluno Beetle 模块，在 Bluno Beetle 中进行解包，得到原始脑电数据，由 Bluno Beetle 将原始脑电信号直接传输至上位机。

4.2. 数据处理

原始信号传输至上位机后，上位机程序内具有特征提取函数及已经预训练的分类模型。在这一阶段，程序会对原始信号进行特征提取，提取出关键特征；接着，这些关键特征会通过已经经过训练的分类模型，对用户的疲劳和清醒状态进行分类。

4.3. 前段数据输出

该设备采用开源 python 函式库 kivy，以设计用户友好的输出界面。上位机通过对应串口对数据进行读取，读取到脑电信号后，可实时输出原始脑电信号。同时，用户还可自主选择输出分类后检测到的清醒/疲劳状态反馈。

5. 作品实物展示



图 10 作品实物图

在使用该设备时，将左侧两个参考电极夹在左右耳垂上，将右侧耳挂置于左耳上，设备即成功佩戴。

6. 作品创新点

1) 采用耳后单通道检测脑电信号，使得用户在佩戴时能够体验到极大的便捷和舒适。

传统的脑机接口设备通常需要使用复杂的传感器系统或者紧贴在头皮上，这样的设计往往不够方便，可能会引起用户的不适。而这款设备仅需放置在脑后单通道处，就能够高效地检测脑电信号，使得检测非常便捷，同时不会给用户带来不适感。这种便捷和舒适性意味着用户可以进行长时间佩戴，从而能够更好的进行疲劳监测。

2) 具有完整的前段交互界面, 功能完整, 不仅能够实时、精确地呈现用户的原始脑电波形, 还通过智能算法分析脑电信号, 实时显示用户的疲劳状态, 为用户提供了一种全新的、便携的疲劳检测方式, 帮助他们更好地了解自身状态, 科学合理地安排生活和工作。

3) 针对过去的疲劳检测模型进行改进, 采用了 CNN+Transformer 的形式。CNN 以其强大的特征提取能力, 使得对于时序信号的特征提取工程可以突破传统方法上的使用滤波与时频转换的手段提取特征。同时, 随着 Transformer 的提出, 其在处理很多序列任务如机器翻译中获得了成功, 传统的序列模型, 如循环神经网络 (RNN) 和卷积神经网络 (CNN), 在处理长期依赖性和全局上下文信息时存在一些限制, 其梯度会受到近处数据的梯度主导, 对于远处的数据的梯度敏感度降低。而 Transformer 则通过引入注意力机制, 能够在不使用任何循环或卷积结构的情况下, 直接对输入序列中的不同位置进行关联和交互, 从而更好地捕捉序列中的依赖关系。Transformer 的关键思想是自注意力机制 (Self-Attention), 它允许模型在生成输出时对输入序列的不同位置进行加权处理, 从而更加灵活地编码序列中的信息。此外, Transformer 还引入了位置编码来捕捉序列中的顺序信息, 并采用了残差连接和层归一化等技术来加强模型的训练和优化过程。利用 CNN 结合 Transformer 的方式可以再提取局部特征的同时提取时序信号的全局依赖性, 使得整个模型的性能得到进一步的升级。

7. 团队分工

何征昊 模型构建

马雪林 硬件设计

杨洁宁 交互设计

黄峥邈 外观设计

8. 未来展望

该作品介绍了一款用于疲劳检测的便携式脑机接口设备。通过检测耳后脑电信号，它能够准确洞察用户的疲劳状态，为个人健康提供了全新的视角。我们的未来发展目标是不断提升这项技术，使其更加精准、可靠。我们致力于开发智能算法，能够更细致地分析用户的脑电信号，从而更精准地判断疲劳程度，甚至预测潜在的身体疲劳风险。同时，我们将持续改进用户界面，使其更加友好直观，让普通用户也能轻松读懂并管理自身健康。此外，我们的目标是推动这项技术走向更广泛的应用领域，包括医疗、体育、工作场所等，为各行各业的人们提供健康咨询和指导。我们相信，这项创新技术将在未来成为个人健康管理的必备工具，为人们创造更健康、更高效的生活方式。