***2025***



**人工智能导论**

**课程报告**

**选题：**

**引入注意力模型对脑电数据进行注意力分析**

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： |  |
| 学 号： |  |
| 姓 名： |  |
| 指导老师： | 莫益军 |
| 完成日期： | 2025.01.01 |

**摘要**

本项目的目的为引入注意力机制对脑电（EEG）数据进行注意力分析，实现了一个结合卷积神经网络（CNN）和注意力机制的注意力分析模型。本报告首先对当前该领域稀少的研究进行了总结，并对脑电数据的特征进行分析，然后详细记录了该模型的实现过程，并对其性能进行了优化。

**关键词：**脑电数据 注意力机制 卷积神经网络 人工智能

**Abstract**

The purpose of this project is to introduce the attention mechanism for task analysis and prediction of electroencephalography (EEG) data, resulting in a task analysis and prediction model that combines convolutional neural networks (CNN) with attention mechanisms. This report first summarizes the scarce research in this field, analyzes the characteristics of EEG data, and then documents the implementation process of the model in detail, along with optimizations to its performance.

**Keywords**：EEG Attention CNN AI

目录

[一、研究背景 4](#_Toc187875215)

[二、研究方法 5](#_Toc187875216)

[1. EEG数据特征分析 5](#_Toc187875217)

[2. 注意力机制 6](#_Toc187875218)

[2. 卷积神经网络 9](#_Toc187875219)

[3. 多层感知机 10](#_Toc187875220)

[三、研究结果 11](#_Toc187875221)

[1. 模型训练 11](#_Toc187875222)

[2. 数据验证 12](#_Toc187875223)

[四、总结与课程感想 13](#_Toc187875224)

[附录：代码 15](#_Toc187875225)

## 一、研究背景

注意力机制（Attention Mechanism）模仿了人类在面对复杂任务时更关注重要信息的行为。自Transformer模型提出以来，在自然语言处理（NLP）和计算机视觉（CV）等领域均有出色的表现。脑电图（EEG）是一种用于记录脑部电活动的非侵入性技术，通过在头皮上放置电极，测量和记录脑细胞的电活动信号，在临床和研究中广泛应用于癫痫诊断、睡眠障碍研究、神经系统疾病检测以及认知功能研究等领域。

除了疾病外，EEG数据的应用还有诸如情绪识别、意图识别与检测、认知评价等领域。然而，传统主流的EEG数据的分析以支持向量机、卷积神经网络和长短期神经网络为主[[1]](#footnote-1)。

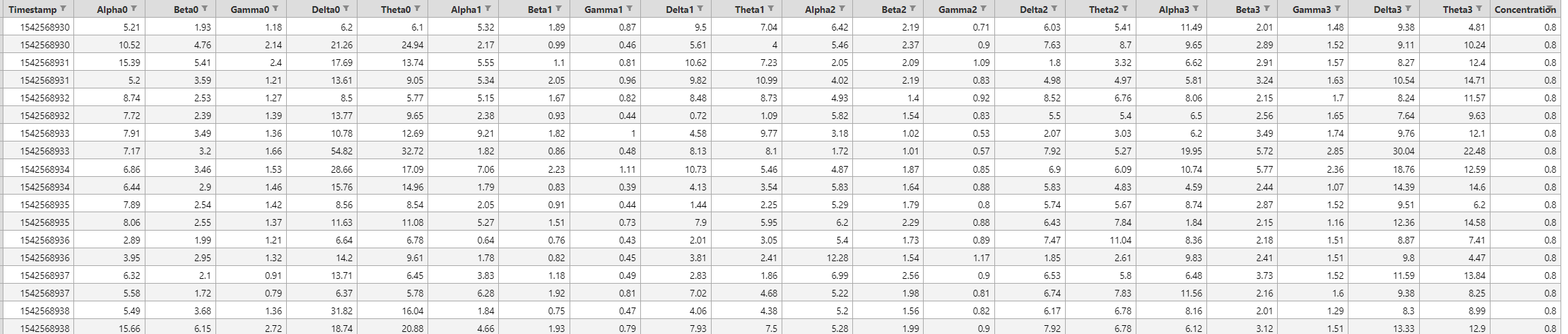
在引入注意力机制的工作上，有研究提出了一种基于注意力机制的卷积循环神经网络（ACRNN）用于情绪识别[[2]](#footnote-2)；有研究采用Transformer模块的编码器-解码器结构，通过堆叠自注意力机制和逐点全连接层提取EEG数据中的信息[[3]](#footnote-3)；有研究提出了一种基于半监督微调自监督图注意力网络（SFT-SGAT）的跨被试脑电（EEG）情感识别和意识检测方法[[4]](#footnote-4)。

本项目基于Kaggle上用户上传的个人脑电数据[[5]](#footnote-5)，结合卷积神经网络和注意力机制，对注意力集中度进行分析预测。

## 二、研究方法

### 1. EEG数据特征分析

本项目使用的数据集是作者使用Muse耳机记录的EEG数据。受试者为男性，录制时年龄为20/21岁。数据包含四个不同电极的α波、β波、γ波、δ波和θ波，并已应用了快速傅里叶变换（128个频率区间）。数据集共计80个CSV文件。

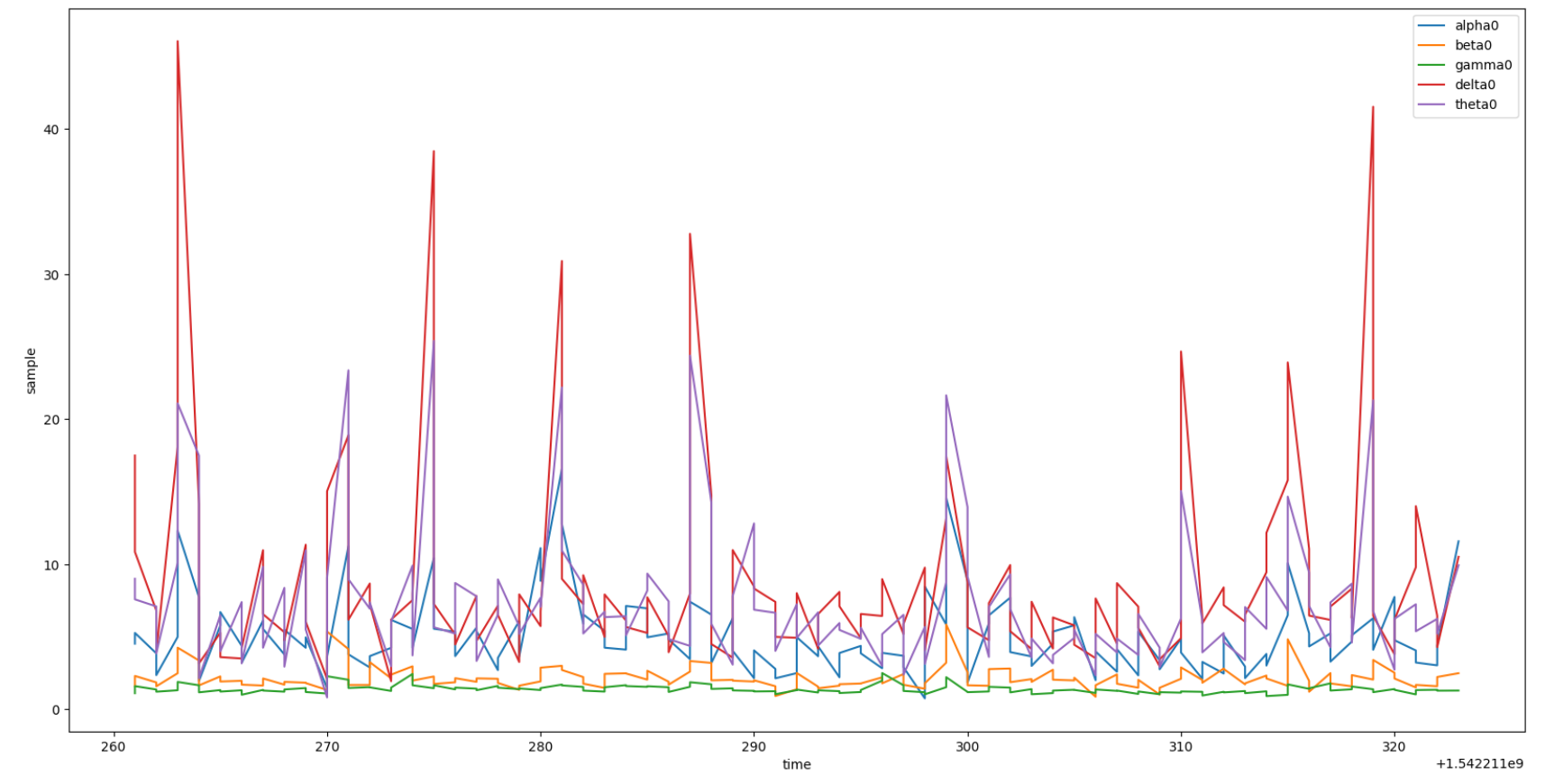


图表 1 1c9nk.csv文件数据示意图

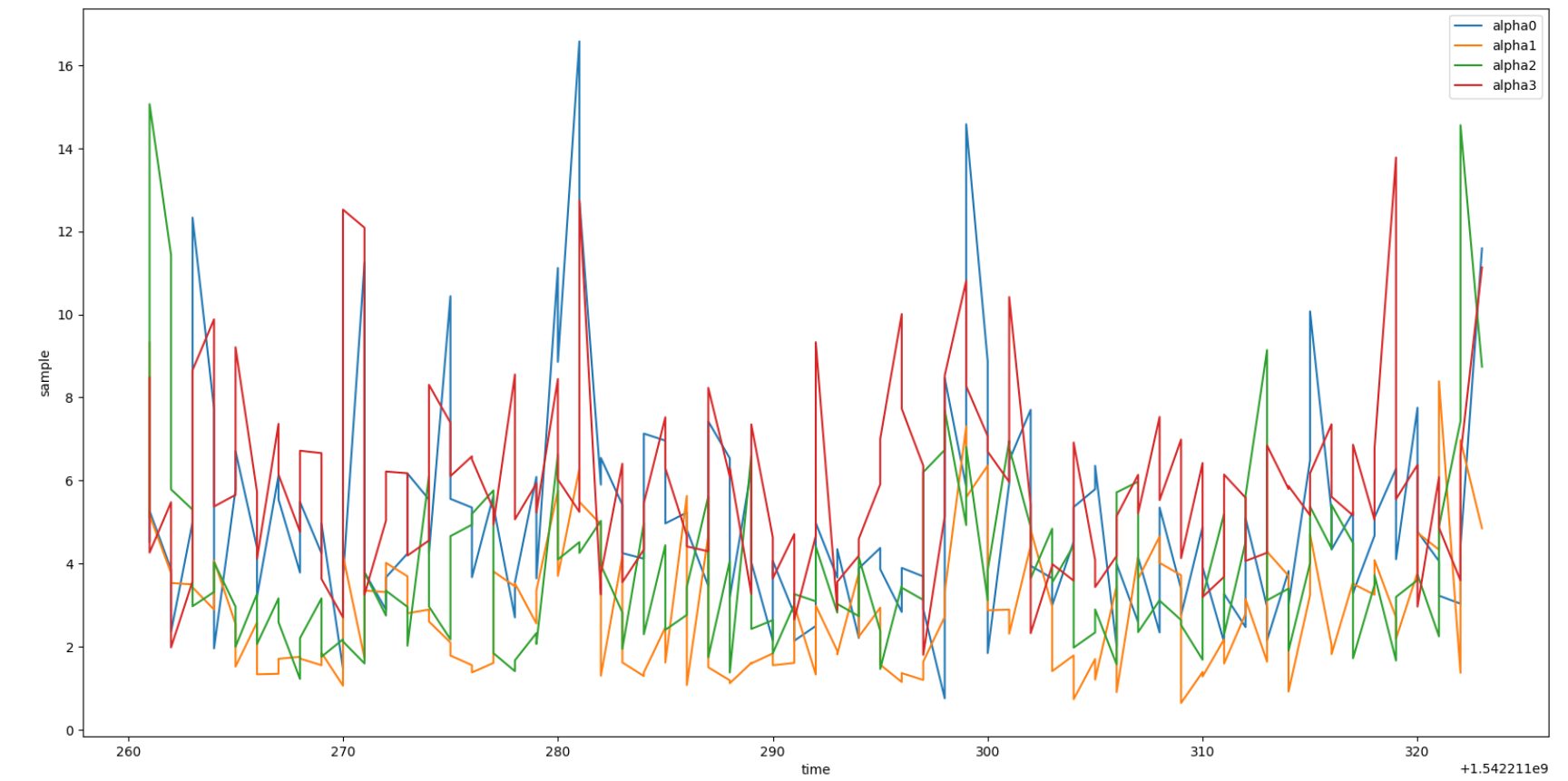
具体数据如上图所示，有如下特征：

1. 数据每一行表示一帧瞬时信息，其中第一列为时间戳，按顺序从上到下排列。
2. 第2至21列共20列为四个不同电极的α波、β波、γ波、δ波和θ波的通道信号数据。
3. 数据最后一列为作者自我评估的注意力集中力值(concentration )，范围在0到1之间，表示注意力的集中程度。

各通道信号数据直观表示如下，可见不同波的幅频区别较明显，而同一种波的不同通道特征较相似：



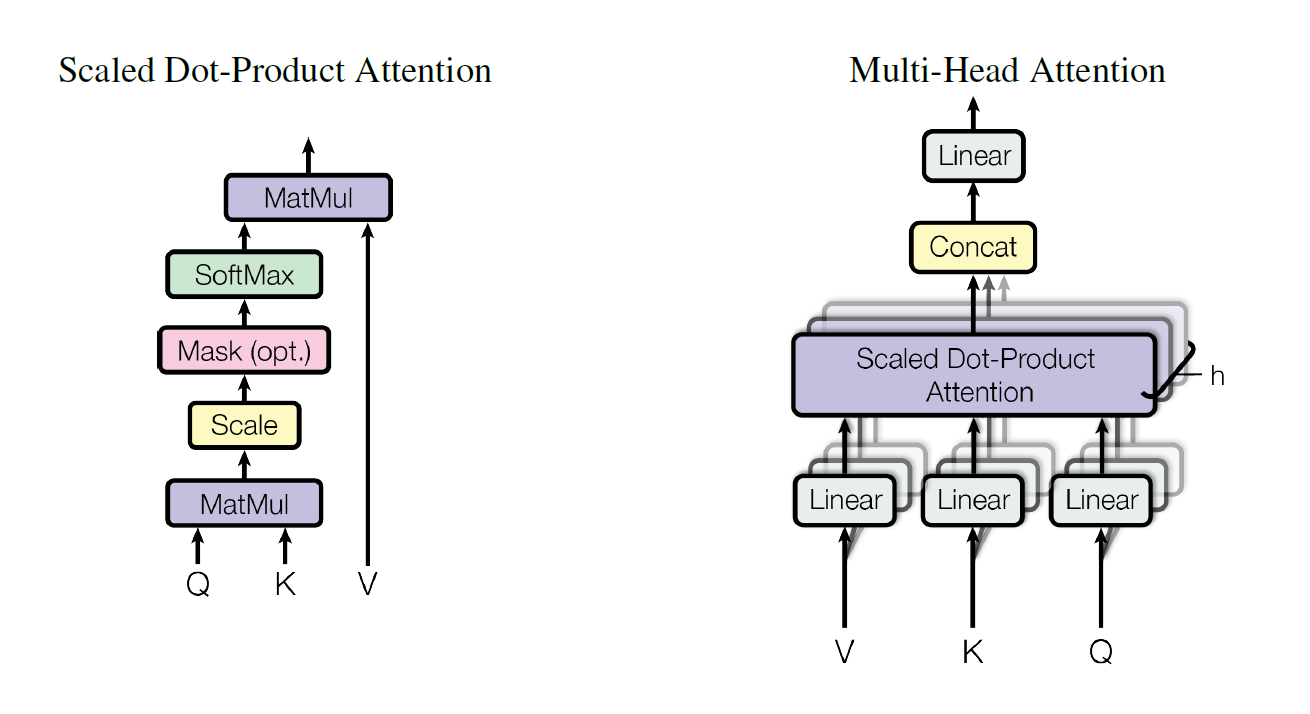
图表 2 不同类型波的图像



图表 3 不同通道下同一类型波的图像

### 2. 注意力机制

经典的注意力机制包括缩放点积注意力和多头注意力。其流程示意图如下图所示：



图表 4 缩放点积注意力与多头注意力机制

缩放点积注意力的基本算法是通过计算查询向量（Query）、键向量（Key）和值向量（Value）之间的点积来衡量相关性，其公式为：

其中，Q是查询矩阵，K是键矩阵，V是值矩阵，是键向量的维度。

* 1. **EEG数据的embedding表示**

在《Attention is All You Need》论文中，首次将注意力机制运用到机器翻译任务中，并取得了优秀的结果。阅读原文，进行机器翻译任务时，首先需要得到各分词的embedding。那么，在我与合作同学讨论时，第一个问题便是：对于EEG数据， embedding是什么？

自然语言处理中使用embedding将一个分词或符号映射到一个高维向量上。分词或符号的含义上越是接近，向量的关系越紧密，表现为向量的方向、长度越是接近，向量间的夹角越小；反之，则两向量夹角越大。

不难发现，EEG各通道数据天然便具有如上特性，20个通道构成的20维向量越是接近，说明此时状态约相似；反之则约相异。因此，我们不妨直接把各帧20个通道的1x20向量作为基础embedding。

在此基础上，我们再对样本加上位置embedding。经典的位置embedding由下列公式计算：

对于序列中第个位置和整数：

1. 维度为
2. 维度为

最后，将基础embedding与位置embedding直接相加，得到EEG数据的embedding表示。

**2.2 Q、K、V矩阵的计算**

为了方便权重矩阵的求解，我们不妨将各数据文件中数据以固定长度如20行为一组（约10s）构成一份样本。则各样本的特征矩阵X的形状为，标签即为各行醉后一列共同的concentration值。

则Q、K、V矩阵的计算为：

其中为待训练的权重矩阵。

那么怎么理解Q、K、V矩阵呢？从人的行为角度分析，我们对复杂事务的关注或注意过程可分为以下三个意识过程：

1. 我该关注的事物有什么性质？
2. 我去何时何地寻找应关注的事物？
3. 该关注的事物的在该问题语境下的表示是什么？

以上三个步骤即对应Q、K、V矩阵的求解，这个过程是数据驱动的。

之后，有了Q、K、V矩阵后，我们便可根据上述缩放点积注意力的公式得到样本的整体特征，从而建立样本与注意力标签的对应关系。

对缩放点积注意力公式的直观理解：

1. 表示，在问题查询下得到了对各个位置应该关注的程度，并将数值缩放处理，维持数值分布的稳定性，避免梯度消失问题。
2. 将结果进行概率分布归一化表示。
3. 表示使用关注程度与各位置的数字表示进行点积操作得到注意力输出，即，关注到应该关注的部分后，得到的样本的整体特征内涵。

通过上述步骤，缩放点积注意力机制能够有效地计算输入样本中每个元素之间的相关性，并生成加权的输出，以供更进一步的数据分析。

### 2. 卷积神经网络

由于EEG数据的时序特性，在使用注意力机制之前，还可以通过卷积神经网络（Convolutional Neural Networks，以下简称CNN）进行初步的特征提取。

由于EEG数据各通道间相对独立，不难想到，对各通道的时序数据作一维卷积处理，从而得到其局部特征，增强模型的表达能力。

本模型中简单使用了一层一维卷积网络：

|  |
| --- |
|  |

### 多层感知机

上述过程中，经过卷积神经网络、注意力机制后，得到加权注意力输出矩阵。那么接下来的问题便是：如何将其映射到注意力数值上呢？

显然我们要建立“多维矩阵→单个数值”的映射。经验上，一维向量到数值是容易的，只需要对向量的各个元素加权求和便可。那么，多维矩阵如何转为一维向量呢？

多维矩阵的降维又可以叫池化（pooling）操作，存在以下几种选择：

1. 直接将矩阵拆分重组为一维矩阵。

优点：操作简便，维持数据信息量。

缺点：计算量大。

1. 最大/最小/全局池化：将矩阵各维元素作最大/最小/平均值处理。

优点：计算简便，数据规模减小。

缺点：损失数据信息。

1. 注意力池化：有选择的将数据进行加权处理。

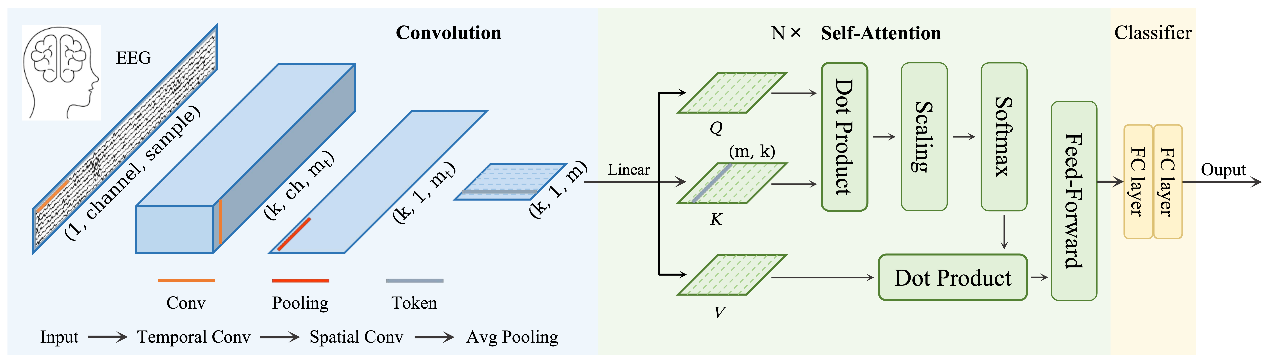
优点：数据规模减小，不损失数据原始信息。

缺点：权重矩阵需要训练得到。

从模型的表达能力等方面综合考虑，本模型中我们选择了注意力池化。

在池化操作之后，我们得到了一维向量。再通过两层全连接层与一层激活层，最终得到注意力值的输出。

本模型的结构示意图为：



图表 5 模型结构示意图

结构的具体为：

|  |
| --- |
| **EEGAttentionModel**(  (conv): Conv1d(20, 100, kernel\_size=(3,), stride=(1,), padding=(1,))  (transformer\_encoder): TransformerEncoder(  (layers): ModuleList(  (0-7): 8 x TransformerEncoderLayer(  (self\_attn): MultiheadAttention(  (out\_proj):NonDynamicallyQuantizableLinear(in\_features=100, out\_features=100, bias=True)  )  (linear1): Linear(in\_features=100, out\_features=128, bias=True)  (dropout): Dropout(p=0.1, inplace=False)  (linear2): Linear(in\_features=128, out\_features=100, bias=True)  (norm1): LayerNorm((100,), eps=1e-05, elementwise\_affine=True)  (norm2): LayerNorm((100,), eps=1e-05, elementwise\_affine=True)  (dropout1): Dropout(p=0.1, inplace=False)  (dropout2): Dropout(p=0.1, inplace=False)  )  )  )  (fc): Sequential(  (0): Linear(in\_features=100, out\_features=128, bias=True)  (1): ReLU()  (2): Linear(in\_features=128, out\_features=1, bias=True)  )  ) |

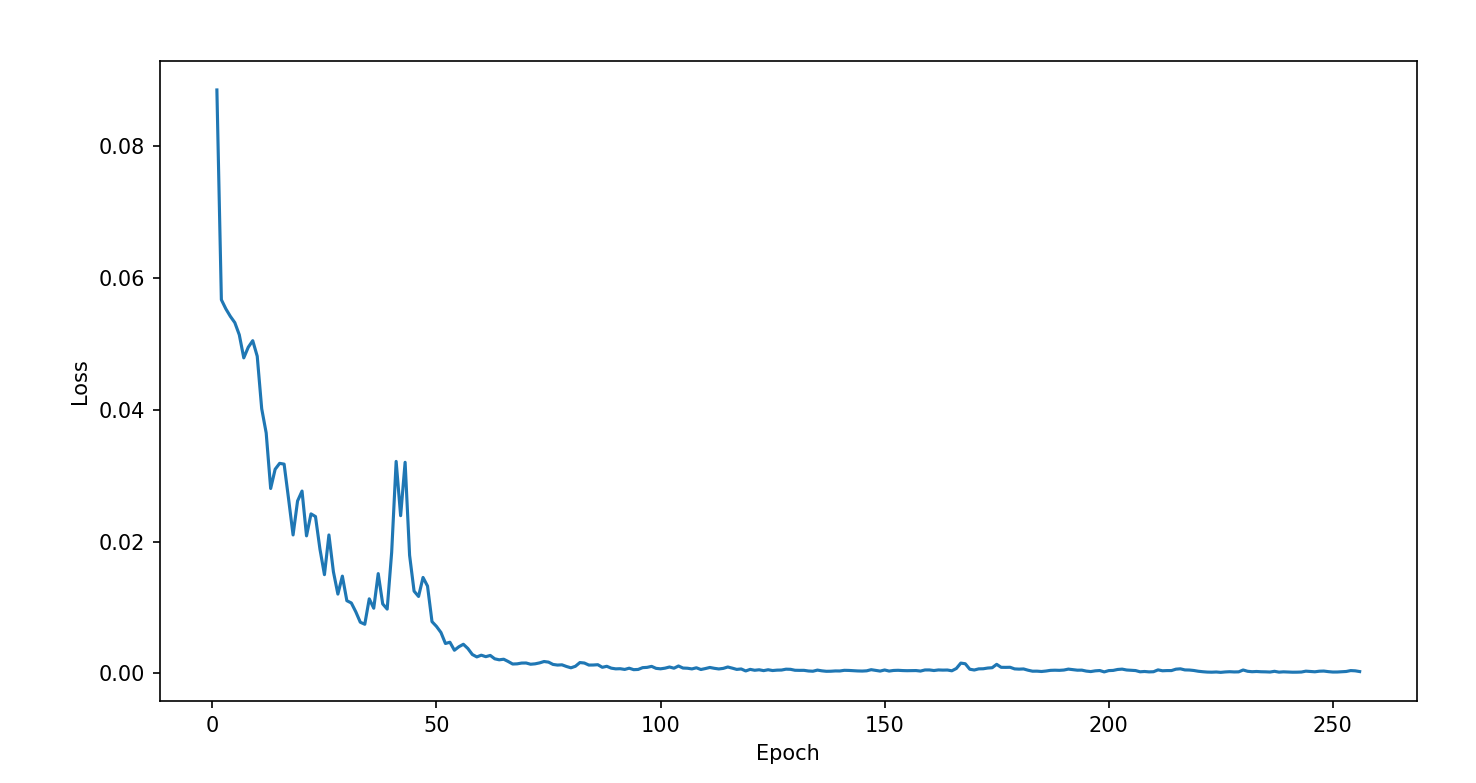
## 三、研究结果

### 1. 模型训练

本模型训练时，选择使用均方根误差（MSE）作为评估函数，选择使用自适应矩阵估计（Adam）作为优化算法，并经大量尝试后，使用了以下超参数：

|  |  |  |
| --- | --- | --- |
| 超参数 | 说明 | 值 |
| embedding\_dim | 控制embedding向量的维度 | 100 |
| num\_heads | 控制注意力头个数 | 32 |
| num\_layers | 控制Transformer编码层数 | 8 |
| hidden\_dim | 控制全连接中间隐藏层维度 | 256 |
| batch\_size | 控制样本批处理大小 | 32 |
| learning\_rate | 控制学习率 | 0.005 |
| num\_epochs | 控制训练伦次 | 256 |

训练过程中的损失变化如下图所示：

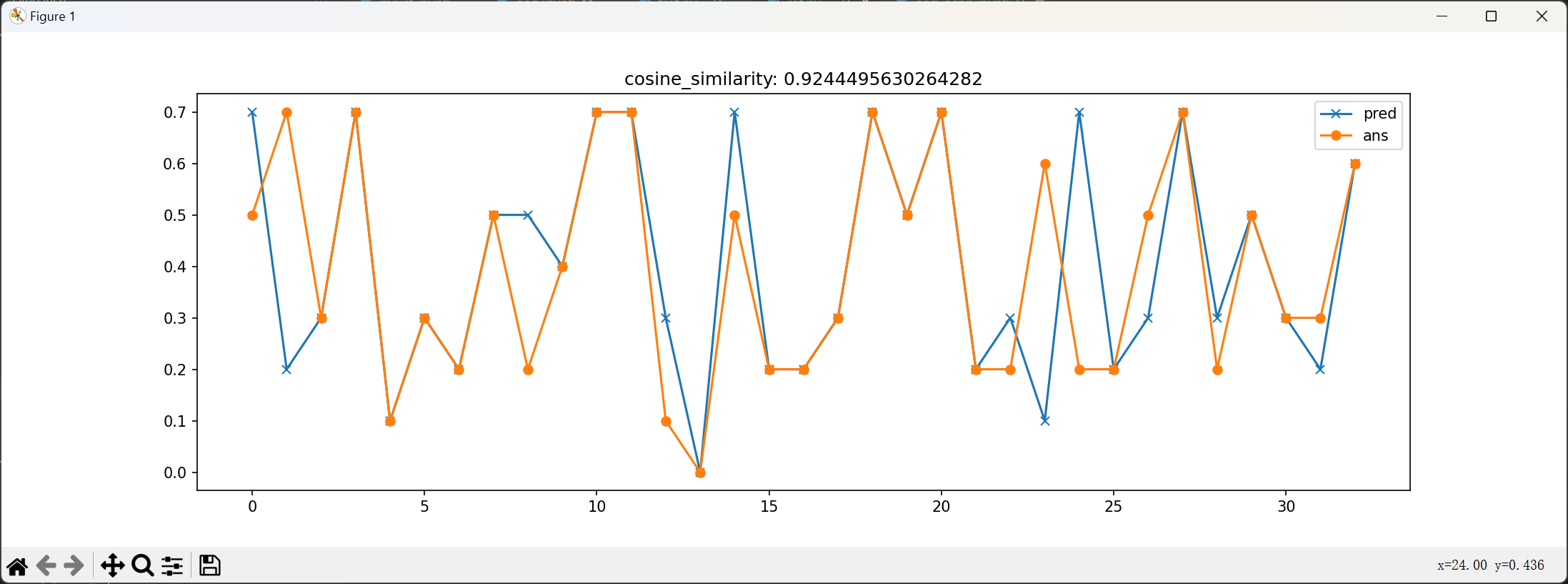


图表 6 训练过程模型损失变化

可见，模型损失在训练过程中呈下降趋势，并最后保持稳定，说明模型收敛良好。

### 数据验证

将数据集按照9：1的比例分为训练集和测试集。在测试集上运行模型得到的结果与标签值对比，如下图所示：



图表 7 数据验证对比图

如图所示可见，模型输出与标准值基本保持一致，余弦相似度达92.4%，说明模型效果良好。

## 四、总结与课程感想

从一开始我便对本次人工智能导论课充满期待。亲眼见证了大模型的问世，不了解其原理的我就像看到神迹一样。大模型就像戴着面纱的美女，朦胧罩着一层神秘的面纱。因此在选题时，我毫不犹豫的选择了它的基石之一——Transformer与注意力机制。

在这之前，我也多次接触过大模型的应用。除了玩乐式的借助工具制作语音生成模型、实现虚拟“声优”以及AI绘图外，还在大创中开发了移动端本地实时场景语音解说应用。印象最深刻的便是如何给网上的模型“喂”数据，为什么要这么“喂”，如何加快模型生成速度，能不能让模型并行生成next token？

在本次课设后，我想我已经明白了。

回顾本次课设过程，一开始我对着论文，感到手足无措。从公众号上找了几篇文章看，仍觉云里雾里。embedding是什么？Q、K、V是什么？为什么要这么做？数据走完注意力机制后，下一步又该怎么映射到单个数值？等等问题给我带来深深的疑惑。好在兴趣是最好的老师，我在整个过程中幸运地始终保持着足够的兴趣，支持我完成整个课程设计。虽然细节方面还有许多模糊之处，但令自己感到愉悦的是抓住了其中的思想。深刻感受到了数据驱动的魅力！

最后，感谢莫老师一学期以来耐心的为我们讲解人工智能的流派、发展和理论，老师的讲课和前沿知识给了我很深的启发。感谢老师，感谢助教，向你们表示最真挚的感谢，让我学到这么有意思的课程！

## 附录：代码

import math

import torch

import os

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

import torch.nn as nn

import torch.optim as optim

import torch.nn.functional as F

from torch.utils.data import Dataset, DataLoader

class EEGAttentionModel(nn.Module):

def \_\_init\_\_(self, embedding\_dim=100, num\_heads=4, num\_layers=8, hidden\_dim=128, dropout\_rate=0.1):

super(EEGAttentionModel, self).\_\_init\_\_()

# 1D CNN层提取时序特征

self.conv = nn.Conv1d(

in\_channels=20, # 20个脑电通道

out\_channels=embedding\_dim,

kernel\_size=3,

padding=1

)

# Transformer编码器

encoder\_layer = nn.TransformerEncoderLayer(

d\_model=embedding\_dim,

nhead=num\_heads,

dim\_feedforward=hidden\_dim,

dropout=dropout\_rate,

batch\_first=True

)

self.transformer\_encoder = nn.TransformerEncoder(

encoder\_layer,

num\_layers=num\_layers

)

# 注意力池化

self.query = nn.Parameter(torch.randn(embedding\_dim))

# 最终预测层

self.fc = nn.Sequential(

nn.Linear(embedding\_dim, hidden\_dim),

nn.ReLU(),

nn.Linear(hidden\_dim, 1),

)

def forward(self, x):

# x shape: [batch, timesteps, channels]

x = x.permute(0, 2, 1) # 转换为[batch, channels, timesteps]

x = self.conv(x) # CNN提取特征

x = x.permute(0, 2, 1) # 准备给Transformer

# Transformer处理序列

x = self.transformer\_encoder(x)

# 注意力池化

scores = torch.einsum('btd,d->bt', x, self.query)

attn = F.softmax(scores, dim=1).unsqueeze(-1)

x = (x \* attn).sum(dim=1)

# 预测concentration

return self.fc(x)

# 数据加载器

class EEGDataset(Dataset):

def \_\_init\_\_(self, path, timesteps=20):

self.data = []

self.labels = []

t = {

0.0: 0,

0.1: 0.1,

0.2: 0.2,

0.4: 0.3,

0.7: 0.4,

0.8: 0.5,

0.9: 0.6,

1.0: 0.7,

}

# 加载CSV文件

for file in list(os.walk(path))[0][2]:

if file.endswith('.csv'):

df = pd.read\_csv(os.path.join(path, file))

# 提取特征和标签

features = df.iloc[:, 1:-1].values # 所有通道数据

labels = df.iloc[:, -1].values # concentration值

# 分段处理

for i in range(0, len(df)-timesteps+1, timesteps):

self.data.append(features[i:i+timesteps])

self.labels.append(t[labels[i]])

def \_\_len\_\_(self):

return len(self.data)

def \_\_getitem\_\_(self, idx):

return torch.FloatTensor(self.data[idx]), torch.FloatTensor([self.labels[idx]])

# 超参数

embedding\_dim = 100

num\_heads = 32

num\_layers = 8

hidden\_dim = 256

batch\_size = 32

learning\_rate = 0.0005

num\_epochs = 256

# 准备数据

dataset = EEGDataset("./dqmonn/personal-eeg-tasks/versions/4/data")

train\_size = int(0.9 \* len(dataset))

test\_size = len(dataset) - train\_size

train\_dataset, test\_dataset = torch.utils.data.random\_split(

dataset, [train\_size, test\_size])

train\_loader = DataLoader(train\_dataset, batch\_size=batch\_size, shuffle=True)

test\_loader = DataLoader(test\_dataset, batch\_size=batch\_size)

# 初始化模型

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

model = EEGAttentionModel().to(device)

criterion = nn.MSELoss()

optimizer = optim.Adam(model.parameters(), lr=learning\_rate)

# 训练代码

def train\_model():

ls = []

# 训练循环

for epoch in range(num\_epochs):

model.train()

total\_loss = 0

for batch\_x, batch\_y in train\_loader:

batch\_x, batch\_y = batch\_x.to(device), batch\_y.to(device)

optimizer.zero\_grad()

outputs = model(batch\_x)

loss = criterion(outputs, batch\_y)

loss.backward()

optimizer.step()

total\_loss += loss.item()

print(f'Epoch {epoch+1}, Loss: {total\_loss/len(train\_loader):.4f}')

ls.append(total\_loss/len(train\_loader))

x = range(1, len(ls)+1)

plt.plot(x, ls)

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.show()

def test\_model():

# 测试模型，计算准确率

model.eval()

total\_loss = 0

with torch.no\_grad():

pred = []

ans = []

for batch\_x, batch\_y in test\_loader:

batch\_x, batch\_y = batch\_x.to(device), batch\_y.to(device)

outputs = model(batch\_x)

# 保存并绘图与答案比较

for i in outputs.cpu().numpy():

pred.append(round(i.item()\*10)/10)

# pred.extend(outputs.cpu().numpy())

ans.extend(batch\_y.cpu().numpy())

x = range(len(pred))

r = torch.cosine\_similarity(torch.Tensor(pred).reshape(1, -1), torch.Tensor(ans).reshape(1, -1))

plt.plot(x, pred, label='pred', marker='x')

plt.plot(x, ans, label='ans', marker='o')

plt.legend()

plt.title(f'cosine\_similarity: {r[0].item()}')

plt.show()

train\_model()

test\_model()

1. Karthiga, M., Suganya, E., Sountharrajan, S. et al. Eeg based smart emotion recognition using meta heuristic optimization and hybrid deep learning techniques. Sci Rep 14, 30251 (2024). [↑](#footnote-ref-1)
2. Tao, W., C. Li, R. Song, J. Cheng, Y. Liu, F. Wan, and X. Chen, EEG-based emotion recognition via channel-wise attention and self attention. IEEE Transactions on Affective Computing, 2020. [↑](#footnote-ref-2)
3. J. Xie et al., "A Transformer-Based Approach Combining Deep Learning Network and Spatial-Temporal Information for Raw EEG Classification," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 30, pp. 2126-2136, 2022, doi: 10.1109/TNSRE.2022.3194600. [↑](#footnote-ref-3)
4. Lina Qiu, Liangquan Zhong, Jianping Li, Weisen Feng, Chengju Zhou, Jiahui Pan,

   SFT-SGAT: A semi-supervised fine-tuning self-supervised graph attention network for emotion recognition and consciousness detection,Neural Networks,Volume 180,2024 [↑](#footnote-ref-4)
5. Dominic Monn, <https://www.kaggle.com/datasets/dqmonn/personal-eeg-tasks> [↑](#footnote-ref-5)