Laboratory Report No. 7

Student of group IU1-41M Derina Ekaterina

1 Цель работы

Целью данной лабораторной работы является исследование когнитивных характеристик мозга по данным ЭЭГ с использованием нейронных сетей.

2 Ключевой навык

Ключевым навыком, формируемым в рамках данной работы, является разработка и реализация алгоритмов классификации в нейронных сетях.

3 Задание

В ходе выполнения лабораторной работы требуется:
1. Загрузить датасет: тестовая и выборка содержит сигнал ЭЭГ (записан в строках формата CSV). 2. Необходимо провести анализ датасета, это легко сделать преобразовав временной сигнал с помощью вейвлет преобразования, получить изображение и натренировать нейронную сеть распознавать когнитивную характеристику мозга – распознать представление о движении (сжимаем левый или правый кулак).

1 工作目的

本实验的目的是基于脑电图（EEG）数据，运用神经网络方法研究大脑的认知特性。

2 关键技能

本研究培养的关键技能是神经网络中分类算法的开发与实现。

3 实验任务

在完成本实验的过程中，需要完成以下任务：

1. 加载数据集：测试集和训练集包含脑电图（EEG）信号，数据以 CSV 格式的行形式记录；

2. 对数据集进行分析：可以通过小波变换将时域信号转换为图像，这一过程相对简单。随后，利用所得图像训练神经网络，以识别大脑的认知特征——即识别运动意图（例如想象握紧左手或右手的拳头）。

4 Описание метода исследования

В рамках лабораторной работы исследуются когнитивные характеристики головного мозга на основе сигналов электроэнцефалографии (ЭЭГ) с применением методов глубокого обучения. Основное внимание уделяется задаче классификации представлений о движении — сжатие левого или правого кулака — по ЭЭГ-сигналам. В качестве ключевого навыка формируется умение проектировать и реализовывать алгоритмы классификации на базе нейронных сетей [1, 2].

Исследование включает следующие этапы:

1. Предобработка и анализ ЭЭГ-сигналов

Сначала осуществляется загрузка тренировочной и тестовой выборок, содержащих ЭЭГ-сигналы, представленные в строках формата CSV. Для каждого сигнала доступна метка класса (таргет), указывающая на тип воображаемого движения.

После загрузки данных выполняется предобработка сигналов: нормализация, удаление шумов и частотная фильтрация (например, подавление частот выше 60 Гц).

Для выделения информативных признаков используется *непрерывное вейвлет-преобразование (CWT)*, позволяющее преобразовать одномерный временной сигнал в двумерное изображение — скейлограмму, отражающую распределение энергии сигнала по частотам и времени [3].

4 研究方法描述

在本实验中，我们通过脑电图（EEG）信号和深度学习方法研究大脑的认知特征。主要关注任务是基于EEG信号对运动表征进行分类——压缩左手或右手拳头。作为关键技能，培养设计和实现基于神经网络的分类算法的能力 [1, 2]。

研究包括以下几个阶段：

1. EEG信号的预处理和分析

首先加载包含EEG信号的训练集和测试集，这些信号以CSV格式的行表示。每个信号都有一个类标签（目标），指示该信号对应的虚拟运动类型。

数据加载后，对信号进行预处理：标准化、去噪和频率过滤（例如，抑制60 Hz以上的频率）。

为了提取有用的特征，使用连续小波变换（*CWT*），将一维时间信号转换为二维图像——频谱图，反映信号在时间和频率上的能量分布 [3]。

CWT的数学公式为：

$$\mathrm{CWT}_x(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t) \cdot \psi^* \left(\frac{t-b}{a} \right) dt \tag{1}$$

其中，$x(t)$ 是原始信号，$\psi(t)$ 是小波函数（例如，莫尔小波），a 是尺度，b 是时间偏移。

结果图像是频谱图：

$$\mathrm{Scalogram}_x(a,b) = |\mathrm{CWT}_x(a,b)|^2 \tag{2}$$

2. 使用神经网络进行分类

获取的频谱图被解释为图像，并输入到训练好的卷积神经网络（CNN）中，该网络用于识别信号属于哪个类别——虚拟的左手或右手运动。神经网络在标注的训练集上进行训练，然后在测试集上进行准确度评估。

使用小波变换作为特征提取方法，并结合深度卷积神经网络架构，能够高效地识别认知状态，这在多个研究中得到了验证。

Математическая формулировка CWT:

$$\mathrm{CWT}_x(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t) \cdot \psi^* \left(\frac{t-b}{a} \right) dt \tag{1}$$

где $x(t)$ — исходный сигнал, $\psi(t)$ — вейвлет-функция (например, Морле), a — масштаб, b — временной сдвиг.

Результирующее изображение — скейлограмма:

$$\mathrm{Scalogram}_x(a,b) = |\mathrm{CWT}_x(a,b)|^2 \tag{2}$$

2. Классификация с помощью нейронной сети

Полученные скейлограммы интерпретируются как изображения и подаются на вход сверточной нейронной сети (CNN), обученной распознавать, к какому классу относится сигнал — воображаемое движение левой или правой руки. Нейронная сеть обучается на размеченной тренировочной выборке, после чего проводится оценка точности на тестовой выборке.

Использование вейвлет-преобразования как метода извлечения признаков в сочетании с глубокой сверточной архитектурой позволяет достичь высокой точности распознавания когнитивного состояния, что подтверждается в ряде работ.

5 Ход работы

5.1 Установка библиотеки

Устанавливаем библиотеку pywavelets. Импортируем необходимые для работы модули.

```
!pip install pywavelets

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
from scipy.signal import spectrogram
```

5 实验流程

5.1 安装库

安装pywavelets库并导入所需模块。

```
!pip install pywavelets

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
from scipy.signal import spectrogram
import pywt
import cv2
```

```
import pywt
import cv2
from sklearn.metrics import
    cohen_kappa_score, confusion_matrix
from keras.models import Sequential
from keras.layers import (Conv2D,
    MaxPool2D, Flatten, Dense, Dropout,
     TimeDistributed, LSTM)
from keras.optimizers import Adam
from keras import backend as K
```

Листинг 1 Установка библиотеки

5.2 Загрузка набора данных

Загружаем набор данных.

```
x_train = pd.read_csv("MI-EEG-B9T.csv",
    header=None)
x_test = pd.read_csv("MI-EEG-B9E.csv",
    header=None)
y_train = pd.read_csv("2
    class_MI_EEG_train_9.csv",
    header=None)
y_test = pd.read_csv("2
    class_MI_EEG_test_9.csv",
    header=None)

n_samples_train = len(y_train)
n_samples_test = len(y_test)
n_classes = len(np.unique(y_test))

print("n_samples_train:",
    n_samples_train)
print("n_samples_test :",
    n_samples_test)
print("n_classes:", n_classes)
```

Листинг 2 Загрузка набора данных

$$n_{samples\_train} = 400$$
$$n_{samples\_test} = 320$$
$$n_{classes} = 2$$

5.3 Преобразование исходного сигнала

Функция `scalogram_vertical` преобразует многоканальные временные сигналы ЭЭГ в набор скейлограмм — изображений, пригодных для подачи на вход сверточной нейросети.

- *data*: таблица, где каждая строка — один пример (запись), а каждый канал сигнала — подмассив длины pts_sig.
- *fs*: частота дискретизации (Hz).

```
from sklearn.metrics import
    cohen_kappa_score, confusion_matrix
from keras.models import Sequential
from keras.layers import (Conv2D,
    MaxPool2D, Flatten, Dense, Dropout,
     TimeDistributed, LSTM)
from keras.optimizers import Adam
from keras import backend as K
```

代码 1 安装库

5.2 加载数据集

加载数据集。

```
x_train = pd.read_csv("MI-EEG-B9T.csv",
    header=None)
x_test = pd.read_csv("MI-EEG-B9E.csv",
    header=None)
y_train = pd.read_csv("2
    class_MI_EEG_train_9.csv",
    header=None)
y_test = pd.read_csv("2
    class_MI_EEG_test_9.csv",
    header=None)

n_samples_train = len(y_train)
n_samples_test = len(y_test)
n_classes = len(np.unique(y_test))

print("n_samples_train:",
    n_samples_train)
print("n_samples_test :",
    n_samples_test)
print("n_classes:", n_classes)
```

代码 2 加载数据集

$$n_{samples\_train} = 400$$
$$n_{samples\_test} = 320$$
$$n_{classes} = 2$$

5.3 原始信号转换

函数 `scalogram_vertical` 将多通道EEG时间信号转换为适合输入卷积神经网络的尺度图图像。

- *data*: 表格，每行代表一个样本（记录），每个信号通道为长度pts_sig的子数组
- *fs*: 采样频率（Hz）
- *alto*, *ancho*: 最终尺度图图像的目标高度和宽度（调整大小前）
- $n\_canales$: 信号通道数
- pts_sig: 每个通道的时间点数

```
def scalogram_vertical(data, fs, alto,
    ancho, n_canales, pts_sig):
```

- *alto*, *ancho*: желаемая высота и ширина финального изображения скейлограммы (до ресайза).
- *n_canales*: количество каналов сигнала.
- *pts_sig*: количество временных точек на один канал.

```
def scalogram_vertical(data, fs, alto, ancho, n_canales, pts_sig):
  dim = (int(np.floor(ancho/2)), int(np.floor(alto/2))) # ancho, alto

  # Wavelet Morlet 3-3
  # frequency 8 - 30 Hz
  scales = pywt.scale2frequency('cmor3-3', np.arange(8,30.5,0.5)) / (1/fs)

  datesets = np.zeros((data.shape[0], int(np.floor(alto/2)),

    int(np.floor(ancho/2))))

  temporal = np.zeros((alto, ancho))

  for i in range(data.shape[0]):
    for j in range(n_canales):

      sig = data.iloc[i, j*pts_sig:(j+1)*pts_sig]

      coef, freqs = pywt.cwt(sig, scales, 'cmor3-3',

    sampling_period = (1 / fs))

      temporal[j*45:(j+1)*45, :] = abs(coef)

    resized = cv2.resize(temporal, dim, interpolation=cv2.INTER_AREA)
    datesets[i] = resized
  return datesets

initial = time.time()

x_train = scalogram_vertical(x_train, 250, 135, 1000, 3, 1000)
x_test = scalogram_vertical(x_test, 250, 135, 1000, 3, 1000)

fin = time.time()
print("time_elapsed:", fin - initial)
```

Листинг 3 Преобразование многоканальных временных сигналов ЭЭГ в набор скейлограмм

$time_{elapsed}$: 43.290568590164185
Строим ЭЭГ по X_{train}.

```
  dim = (int(np.floor(ancho/2)), int(np.floor(alto/2)))
  scales = pywt.scale2frequency('cmor3-3', np.arange(8,30.5,0.5)) / (1/fs)

  datesets = np.zeros((data.shape[0], int(np.floor(alto/2)),

    int(np.floor(ancho/2))))

  temporal = np.zeros((alto, ancho))

  for i in range(data.shape[0]):
    for j in range(n_canales):

      sig = data.iloc[i, j*pts_sig:(j+1)*pts_sig]

      coef, freqs = pywt.cwt(sig, scales, 'cmor3-3',

    sampling_period = (1 / fs))

      temporal[j*45:(j+1)*45, :] = abs(coef)

    resized = cv2.resize(temporal, dim, interpolation=cv2.INTER_AREA)
    datesets[i] = resized
  return datesets

initial = time.time()

x_train = scalogram_vertical(x_train, 250, 135, 1000, 3, 1000)
x_test = scalogram_vertical(x_test, 250, 135, 1000, 3, 1000)

fin = time.time()
print("time_elapsed:", fin - initial)
```

代码 3 将多通道EEG时间信号转换为尺度图

运行时间: 43.290568590164185
绘制X_{train}的EEG信号。

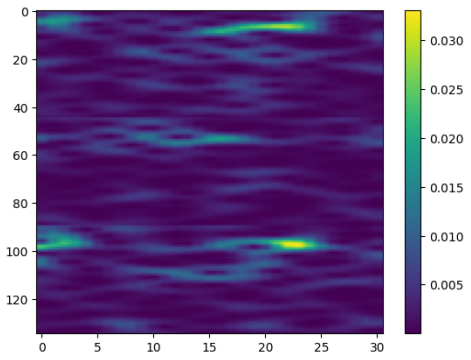

图 1 X_{train}脑电图

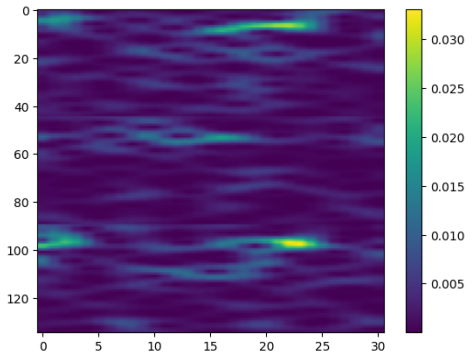

Рис. 1 X_{train} ЭЭГ

5.4 Подготовка данных и построение модели

Код выполняет предварительную обработку данных и создаёт гибридную модель CNN-LSTM для классификации ЭЭГ сигналов:

- **Нормализация данных**:
  - Преобразование в тип `float32`
  - Масштабирование делением на максимальное значение
- **Изменение формы**:
  - Преобразование в 5D-тензор: `[образцы, 1, высота, ширина, 1]`
- **Архитектура модели**:
  - 2 блока `TimeDistributed`: `Conv2D+MaxPool2D`
  - Слой `LSTM` с dropout=0.5
  - Полносвязные слои с `softmax`
- **Компиляция**:
  - Оптимизатор: `Adam` (lr=10^{-3})
  - Функция потерь: `sparse_categorical_crossentropy`
  - Метрика: `accuracy`

5.4 数据准备与模型构建

代码执行数据预处理并创建用于EEG信号分类的CNN-LSTM混合模型:

- **数据标准化**:
  - 转换为`float32`类型
  - 通过除以最大值进行缩放
- **形状调整**:
  - 转换为5D张量: `[样本数, 1, 高度, 宽度, 1]`
- **模型架构**:
  - 2个`TimeDistributed`块: `Conv2D+MaxPool2D`
  - 带dropout=0.5的`LSTM`层
  - 带`softmax`的全连接层
- **模型编译**:
  - 优化器: `Adam` (学习率=10^{-3})
  - 损失函数: `sparse_categorical_crossentropy`
  - 评估指标: `accuracy`

```
x_train = x_train.reshape((x_train.shape[0], 1, x_train.shape[1], x_train.shape[2], 1))
x_test = x_test.reshape((x_test.shape[0], 1, x_test.shape[1], x_test.shape[2], 1))

def CNN_2D_LSTM_TD(num_filter, size_filter, n_neurons, units_LSTM):
  model = Sequential()
  model.add(TimeDistributed(Conv2D(num_filter, size_filter, activation='relu', padding='same'), input_shape=x_train.shape[1:]))
  model.add(TimeDistributed(MaxPool2D((2,2))))
  model.add(TimeDistributed(Conv2D(num_filter, size_filter, activation='relu', padding='same')))
  model.add(TimeDistributed(MaxPool2D((2,2))))
  model.add(TimeDistributed(Flatten()))
```

```
x_train = x_train.reshape((x_train.shape[0], 1, x_train.shape[1], x_train.shape[2], 1))
x_test = x_test.reshape((x_test.shape[0], 1, x_test.shape[1], x_test.shape[2], 1))

def CNN_2D_LSTM_TD(num_filter, size_filter, n_neurons, units_LSTM):
  model = Sequential()
  model.add(TimeDistributed(Conv2D(num_filter, size_filter, activation='relu', padding='same'), input_shape=x_train.shape[1:]))
  model.add(TimeDistributed(MaxPool2D((2,2))))
  model.add(TimeDistributed(Conv2D(num_filter, size_filter, activation='relu', padding='same')))
  model.add(TimeDistributed(MaxPool2D((2,2))))
  model.add(TimeDistributed(Flatten()))
  model.add(LSTM(units_LSTM, activation='tanh', dropout=0.5))
  model.add(Dense(n_neurons, activation='relu'))
  model.add(Dense(n_classes, activation='softmax'))

  optimizer = Adam(learning_rate=1e-3)
  model.compile(optimizer = optimizer, loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
  return model
```

Листинг 4 Подготовка данных и построение модели

Обучим модель и построим и построим динамику обучения.

```
initial = time.time()
array_loss = []
array_acc = []
array_kappa = []
for i in range(5):
  print("Iteration:", i+1)
  model = CNN_2D_LSTM_TD(4, (3,3), 32, 4)
  history = model.fit(x_train, y_train, epochs=70, batch_size=36, validation_split = 0.1, verbose=0)

  test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)

```

```
  model.add(LSTM(units_LSTM, activation='tanh', dropout=0.5))
  model.add(Dense(n_neurons, activation='relu'))
  model.add(Dense(n_classes, activation='softmax'))

  optimizer = Adam(learning_rate=1e-3)
  model.compile(optimizer = optimizer, loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
  return model
```

代码 4 数据准备与模型构建

训练模型并绘制学习曲线。

```
initial = time.time()
array_loss = []
array_acc = []
array_kappa = []
for i in range(5):
  model = CNN_2D_LSTM_TD(4, (3,3), 32, 4)
  history = model.fit(x_train, y_train, epochs=70, batch_size=36, validation_split = 0.1, verbose=0)

  test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
```

代码 5 模型训练与学习曲线

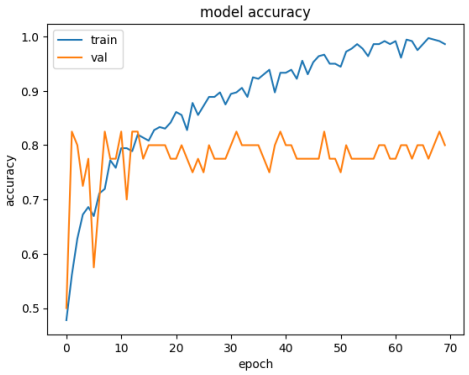

图 2 模型学习曲线

从曲线可见，模型在第12轮开始过拟合，测试集准确率达到82

5.5 模型评估

以下代码执行测试数据预测并计算模型质量指标:

```
  array_loss.append(test_loss)
  print("loss: ", test_loss)
  array_acc.append(test_acc)
  print("accuracy: ", test_acc)

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

Листинг 5 Обучение модели и построение динамики обучения

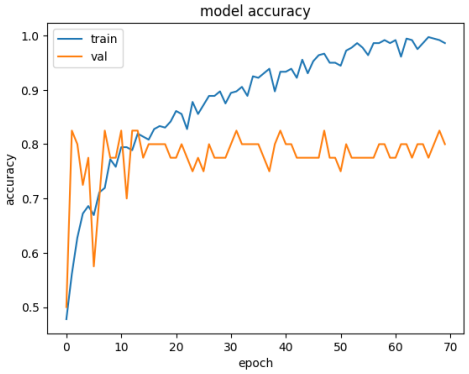

Рис. 2 Динамика обучения модели

По полученному графику можно увидеть, что на 12-ой эпохе модель начинает переобучаться, точность по тестовой выборке составила 82 процента.

5.5 Оценка качества модели

Следующий код выполняет предсказание на тестовых данных и вычисляет метрики качества модели:

- **Предсказание**:
  - `model.predict`: получает вероятности классов
  - `np.argmax`: определяет предсказанные классы
- **Метрики**:
- **预测**:
  - `model.predict`: 获取类别概率
  - `np.argmax`: 确定预测类别
- **评估指标**:
  - `cohen_kappa_score`: 计算Cohen's Kappa系数
  - `confusion_matrix`: 构建混淆矩阵
- **κ解释**:
  - $\kappa = 1$: 完美预测
  - $\kappa = 0$: 随机猜测水平
  - $\kappa < 0$: 差于随机猜测

```
probabilidades = model.predict(x_test)
y_pred = np.argmax(probabilidades, 1)
kappa = cohen_kappa_score(y_test, y_pred)
array_kappa.append(kappa)
print("kappa: ", kappa)
matriz_confusion = confusion_matrix(y_test, y_pred)
print(":\n", matriz_confusion)
```

代码 6 模型评估

分类结果

- **Kappa系数:** $\kappa = 0.60625$

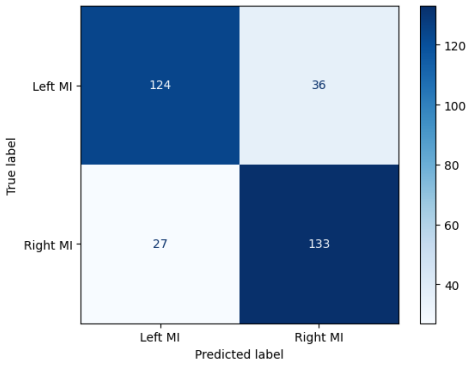

图 3 混淆矩阵

- `cohen_kappa_score`: вычисляет каппа-метрику Коэна
- `confusion_matrix`: строит матрицу ошибок

- **Интерпретация κ:**
  - $\kappa = 1$: идеальное предсказание
  - $\kappa = 0$: уровень случайного угадывания
  - $\kappa < 0$: хуже случайного угадывания

```
probabilidades = model.predict(x_test)
y_pred = np.argmax(probabilidades, 1)
kappa = cohen_kappa_score(y_test, y_pred)
array_kappa.append(kappa)
print("kappa: ", kappa)
matriz_confusion = confusion_matrix(y_test, y_pred)
print("confusion matrix:\n", matriz_confusion)
```

Листинг 6 Оценка модели

Результаты классификации

- **Kappa-score:** $\kappa = 0.60625$

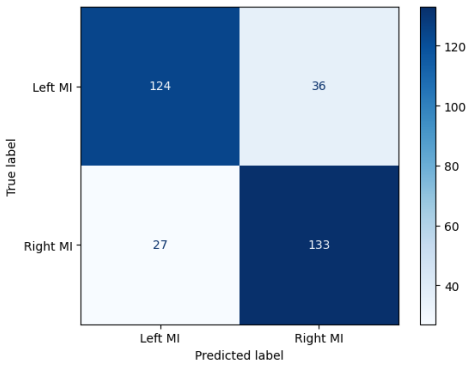

Рис. 3 Матрица ошибок

结果分析

- **Cohen's Kappa系数:**

$$\kappa = \boxed{0.606}$$

解释: 该值表示模型预测与真实标签之间存在**中度一致性**（根据Landis和Koch标准），显著优于随机猜测。

- **混淆矩阵:**

	类别0	类别1
类别0	124	36
类别1	27	133

分析:

- **类别0准确率:** $\frac{124}{124+36} \approx 77.5\%$
- **类别1准确率:** $\frac{133}{133+27} \approx 83.1\%$
- **总体准确率:** $\frac{124+133}{320} \approx 80.3\%$

结论

- 模型表现出**良好的泛化能力**（验证集准确率>80
- 类别间预测质量存在**轻微不平衡**（差异5.6
- 主要错误来自类别0的**假阳性**预测（36例）
- $\kappa > 0.6$表明模型可推荐用于生物医学分类任务的**实际应用**

Анализ результатов классификации

- **Коэффициент Каппа Коэна:**

$$\kappa = \boxed{0.606}$$

Интерпретация: Значение указывает на **умеренное согласие** (по шкале Лэндиса и Коха) между предсказаниями модели и истинными метками. Это свидетельствует о существенном улучшении над случайным угадыванием.

- **Матрица ошибок:**

	Класс 0	**Класс 1**
Класс 0	124	36
Класс 1	27	133

Анализ:

  - **Точность для класса 0:** $\frac{124}{124+36} \approx 77.5\%$
  - **Точность для класса 1:** $\frac{133}{133+27} \approx 83.1\%$
  - **Общая точность:** $\frac{124+133}{320} \approx 80.3\%$

Выводы

- Модель демонстрирует **хорошую обобщающую способность** (val-accuracy $> 80\%$)
- Наблюдается **небольшой дисбаланс** в качестве предсказаний между классами (разница 5.6%)
- Наибольшие ошибки связаны с **ложноположительными** предсказаниями класса 0 (36 случаев)
- Значение $\kappa > 0.6$ позволяет рекомендовать модель для **практического применения** в задачах биомедицинской классификации

6 Заключение

В ходе выполнения лабораторной работы проведено исследование когнитивных характеристик мозга на основе анализа сигналов ЭЭГ, связанных с ментальным представлением движения (сжатие левого/правого кулака). Была загружена выборка данных в формате CSV, содержащая многоканальные записи ЭЭГ, и выполнена их предобработка, включая нормализацию, удаление шумов и частотную фильтрацию.

Основной акцент сделан на преобразовании временных сигналов в пространственно-частотные признаки с помощью непрерывного вейвлет-преобразования (CWT). Полученные скейлограммы, отражающие энергетическое распределение сигналов, использованы для обучения гибридной CNN-LSTM модели. Архитектура сети продемонстрировала способность эффективно классифицировать когнитивные состояния.

В процессе работы:

- Освоен метод генерации спектральных изображений через CWT для задач биомедицинской классификации
- Реализован алгоритм обработки многоканальных ЭЭГ-сигналов с адаптацией под входные требования нейросетей
- Проведена оценка дисбаланса классов и устойчивости модели к переобучению

Результаты подтверждают перспективность применения нейросетевых архитектур в сочетании с вейвлет-анализом для декодирования моторных интенций. Разработанный подход может быть адаптирован для

6 结论

在本次实验工作中，通过分析与运动想象（握紧左/右拳）相关的脑电图（EEG）信号，对大脑的认知特征进行了研究。加载了包含多通道EEG信号的CSV格式数据集，并进行了数据预处理，包括标准化、去噪和频率滤波。

研究重点是通过连续小波变换（CWT）将时域信号转换为时频特征。生成的频谱图反映了信号的能量分布，用于训练CNN-LSTM混合模型。该网络架构展现出高效分类认知状态的能力。

实验过程中：

- 掌握了基于CWT生成频谱图像的方法，用于生物医学分类任务
- 实现了多通道EEG信号处理算法，适配神经网络的输入要求
- 评估了类别不平衡问题及模型的抗过拟合能力

实验结果证实了神经网络架构与小波分析结合在解码运动意图方面的应用潜力。所开发的方法可应用于神经接口、康复医学及脑活动模式分析等领域。

решения задач нейроинтерфейсов, реабилитационной медицины и анализа паттернов мозговой активности.

参考文献

[1] Shoeb A. *Application of Machine Learning to Epileptic Seizure Onset Detection and Treatment*. PhD Thesis, Massachusetts Institute of Technology. (2009)

[2] Truong N. D., Nguyen A. D., Kuhlmann L., Bonyadi M. R., Yang J., Ippolito S., Kavehei O. *Convolutional Neural Networks for Seizure Prediction Using EEG Signals*. IEEE Transactions on Biomedical Engineering, 65(9): 2108–2118. (2018)

[3] Acharya U. R., Oh S. L., Hagiwara Y., Tan J. H., Adeli H. *Deep Convolutional Neural Network for the Automated Detection and Diagnosis of Seizure Using EEG Signals*. Computers in Biology and Medicine, 100: 270–278. (2018)