**Trường Đại học Công Nghệ**

**Viện Trí tuệ nhân tạo**

****

**BÁO CÁO BCI MINI - PROJECT**

**ĐỀ TÀI: MOTOR IMAGE**

**Sinh viên thực hiện:**

### Nguyễn Quang Huy-22022582

**Hà Nội, 12/2024**

# **MỤC LỤC**

1. Mô tả về motor imagery
2. Khái niệm
3. Cơ sở khoa học
4. Ứng dụng
5. Chương trình phân loại
6. Dataset
7. Cài đặt
8. Kết quả
9. Kết luận

# **MỞ ĐẦU**

Trong lĩnh vực giao diện não-máy tính (Brain-Computer Interface - BCI), motor imagery (MI) đã trở thành một chủ đề nghiên cứu đáng quan tâm nhờ tiềm năng ứng dụng cao trong việc tăng cường khả năng giao tiếp, phục hồi chức năng và hỗ trợ người khuyết tật. Motor imagery là quá trình người tham gia tưởng tượng việc thực hiện một hành động vận động cụ thể mà không cần thực sự chuyển động. Tín hiệu điện não được ghi nhận trong quá trình này cung cấp dữ liệu quý giá để giải mã các mệnh lệnh não bộ điều khiển.

Tuy nhiên, phân loại tín hiệu MI là một bài toán thách thức do bản chất phi tuyến tính, nhiễu và biến đổi đàn hồi theo thời gian của tín hiệu EEG. Các tiến bộ gần đây trong lĩnh vực học sâu (deep learning) đã cung cấp những công cụ hữu hiệu cho việc phân tích và phân loại các tín hiệu phức tạp này.

Báo cáo này tập trung vào việc nghiên cứu các phương pháp phân loại tín hiệu MI, từ quy trình tiền xử lý dữ liệu, lựa chọn đặc trưng, đến việc áp dụng ba phương pháp học sâu: Convolutional Neural Networks (CNN), Transformer và Multi-Layer Perceptron (MLP) để phân loại tín hiệu motor imagery. Kết quả nghiên cứu góp phần minh chứng tiềm năng ứng dụng thực tế của các phương pháp phân loại trong hệ thống BCI.

# **Mô tả về motor imagery**

## *Khái niệm*

* Theo wikipedia, Motor imagery (MI) (tưởng tượng vận động) được định nghĩa là trạng thái động trong đó cá nhân kích thích về mặt thần kinh một hành động vật lý. Hiện tượng này được hiểu là cá nhân cảm nhận được họ thực hiện hành động.
* MI là quá trình cá nhân tưởng tượng, hình dung ra hành động mà họ cần thực hiện mà không thực sự thực hiện hành động đó. Ví dụ như khi một người tưởng tưởng việc họ nâng cánh tay phải mà không cử động thực tế.
* MI cần một bộ phận cơ thể hoặc cả cơ thể tham gia vào quá trình tưởng tượng, chẳng hạn như cánh tay trong ví dụ trên, hoặc cả cơ thể khi tưởng tượng đang chạy.
* Tín hiệu từ MI được sử dụng trong BCI do não bộ phát ra tín hiệu tương tự với hành động mong muốn thực hiện khi cá nhân tưởng tượng ra hành động đó. Tín hiệu này có thể quan sát được qua các phương thức neural imagery.

## *Cơ sở khoa học*

* Motor Imagery là một phần của cơ chế biểu diễn hành động (motor representation). Motor representation bao gồm việc dự định và chuẩn bị hành động.
* Quá trình biểu diễn hành động thường diễn ra một cách không có ý thức. Và có thể thực hiện một cách có ý thức khi cá nhân chủ động tưởng tượng ra hành động cần thực hiện.

## *Ứng dụng*

* Motor imagery (MI) có nhiều ứng dụng quan trọng, đặc biệt trong các lĩnh vực liên quan đến thần kinh học, phục hồi chức năng, và công nghệ giao diện não - máy (BCI).
* Ứng dụng của việc phân tích tín hiệu điện não tác vụ tưởng tượng vận động có thể dùng trong điều khiển xe lăn, các hệ thống đánh vần ký tự hoặc phục hồi chức năng bệnh nhân bị suy yếu chức năng vận động.

# **Chương trình phân loại**

## *Dataset*

1.1 Giới thiệu

* Tập dữ liệu dùng trong dự án có nguồn PhysioNet EEG Motor Imagery Dataset.
* PhysioNet EEG Motor Imagery Dataset là một tập dữ liệu nổi bật trong lĩnh vực giao diện não - máy (BCI), tập trung vào việc nghiên cứu các tín hiệu điện não đồ (EEG) liên quan đến Motor Imagery (MI).
* Dữ liệu được thu thập từ 109 người tham gia với nhiều trạng thái hoạt động khác nhau, từ tưởng tượng cho đến thực hiện các động tác nắm tay trái, phải hoặc cả hai tay.

1.2 Cấu trúc của bộ dữ liệu

* Dữ liệu bao gồm các hoạt động liên quan đến motor imagery, được phân loại thành các trạng thái cụ thể:
* **BASELINE\_EYE\_OPEN**: Thư giãn, mở mắt.
* **BASELINE\_EYE\_CLOSED**: Thư giãn, nhắm mắt.
* **OPEN\_CLOSE\_LEFT\_RIGHT\_FIST**: Mở và đóng tay trái/phải.
* **IMAGINE\_OPEN\_CLOSE\_LEFT\_RIGHT\_FIST**: Tưởng tượng mở và đóng tay trái/phải.
* **OPEN\_CLOSE\_BOTH\_FIST**: Mở và đóng cả hai tay.
* **IMAGINE\_OPEN\_CLOSE\_BOTH\_FIST**: Tưởng tượng mở và đóng cả hai tay
* Trong dự này, chỉ sử dụng các trạng thái **IMAGINE\_OPEN\_CLOSE\_LEFT\_ RIGHT\_FIST**, tập trung vào việc tưởng tượng nắm tay trái hoặc tay phải.
* Định dạng và cấu trúc dữ liệu:
* Tín hiệu EEG được lưu dưới dạng các tệp EDF (Electroencephalography Data Format).
* Các nhãn sự kiện tương ứng với hoạt động được lưu dưới dạng chú thích trong mỗi tệp dữ liệu.
* Số lượng người tham gia được chọn: 79 (ID từ 1 đến 79).
* Đặc điểm tín hiệu EEG:
* Dữ liệu EEG được thu từ nhiều kênh, bao gồm:
* Kênh EEG: đo hoạt động điện từ vỏ não.
* Kênh EOG: theo dõi chuyển động mắt.
* Tần số lấy mẫu tín hiệu: đủ cao để bắt các dao động não từ tần số thấp đến cao (alpha, beta, gamma).

## *Code:*[*https://colab.research.google.com/drive/1Ud-SSaQH\_ELdWlL8Ud5sLA7qtiURhrrR?usp=sharing*](https://colab.research.google.com/drive/1Ud-SSaQH_ELdWlL8Ud5sLA7qtiURhrrR?usp=sharing)

## *Cài đặt*

### 2.1 Thư viện

Các thư viện được sử dụng trong dự án như:

* Thư viện xử lý tín hiệu EEG: mne (sử dụng module: concatenate\_raws).
* Thư viện quản lý và tiền xử lý: os, re, io, cv2, string, random, warnings.
* Thư viện tính toán và đồ thị: numpy, matplotlib.pyplot, google.colab.patches.cv2 \_imshow.
* Thư viện học sâu: torch, torch.nn, torch.optim, torch.utils.data, torch.nn.functional, torch.autograd.Variable.
* Thư viện Lightning: lightning (Pytorch Lightning) (sử dụng các module như : trainer, loggers,...)
* Thư viện TorchMetrics : torchmetrics.classification.Accuracy.

Các thư viện trên giúp ta xử lý tín hiệu EEG, xây dựng mô hình học sâu (sử dụng PyTorch), và huấn luyện mô hình với các kỹ thuật tiên tiến từ Lightning trong dự án.

### 2.2 Xử lý dữ liệu

#### 2.2.1 Tải và ghép dữ liệu:

* Dữ liệu từ 79 người tham gia được tải xuống thông qua thư viện **MNE**.
* Các tín hiệu từ các phiên liên quan đến tưởng tượng motor imagery được đọc và ghép lại thành một tập dữ liệu lớn.

#### 2.2.2 Tiền xử lý tín hiệu EEG:

* **Trích xuất tín hiệu thô**: Lấy dữ liệu từ các kênh EEG, loại bỏ các tín hiệu không liên quan (EOG hoặc kênh bị nhiễu).
* **Phân đoạn dữ liệu (Epoching)**:
* Tín hiệu được chia thành các đoạn (epoch) theo thời gian, từ **1 giây đến 4.1 giây** sau khi sự kiện xảy ra.
* Mỗi epoch chứa tín hiệu EEG của một lần thực hiện hoặc tưởng tượng động tác.
* **Chuẩn hóa tín hiệu**: Tín hiệu được chuyển từ volt sang microvolt (V) và lưu dưới dạng số thực 32-bit để giảm độ phức tạp.

#### 2.2.3 Gán nhãn:

* Mỗi thử nghiệm được gán nhãn:
* **0**: Tưởng tượng nắm tay trái.
* **1**: Tưởng tượng nắm tay phải.
* Tổng hợp nhãn vào mảng y để phục vụ các bài toán phân loại.

#### 2.2.4 Đầu ra dữ liệu: Dữ liệu tín hiệu EEG (X)

* Dạng ma trận 3 chiều:
* **Số mẫu (samples)**: Số lần thực hiện tưởng tượng động tác.
* **Số kênh (channels)**: Số lượng kênh EEG được sử dụng.
* **Thời gian (timepoints)**: Số điểm dữ liệu trong mỗi epoch.
* Đơn vị tín hiệu: microvolt (μV).
* Định dạng dữ liệu: float32

#### 2.2.5 Nhãn (y)

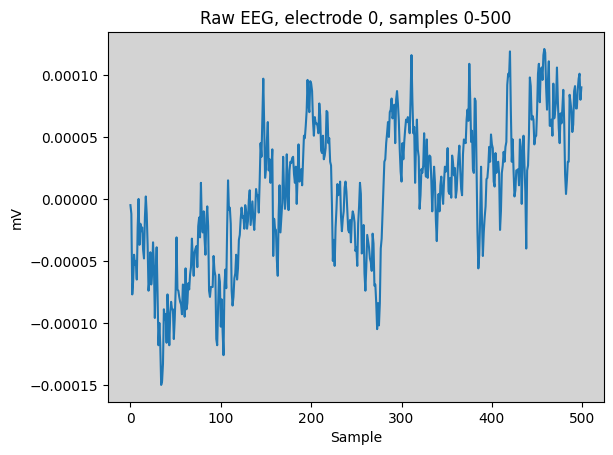
* Một vector 1 chiều, biểu diễn nhãn của từng thử nghiệm:
* 0: Tưởng tượng nắm tay trái.
* 1: Tưởng tượng nắm tay phải.
* Định dạng dữ liệu: int64.

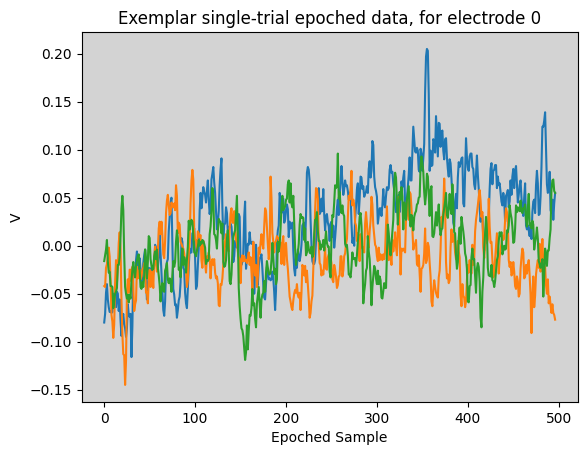
#### 2.2.6 Chia dữ liệu:

* Tách dữ liệu đầu vào **x** (tín hiệu EEG) và nhãn **y** (nếu có) thành 3 tập:
* **train\_ds**: Tập huấn luyện (81%)
* **val\_ds**: Tập kiểm định (9%)
* **test\_ds**: Tập kiểm tra (10%)

#### 2.2.7 Vẽ biểu đồ:

* Tín hiệu EEG thô:



* Tín hiệu EEG thô từ 500 mẫu đầu tiên của điện cực đầu tiên (electrode 0).
* Trục tung (mV): Biên độ tín hiệu (đơn vị millivolt).
* Trục hoành: Chỉ số mẫu (sample index).
* Biểu đồ giúp ta quan sát tín hiệu EEG chưa qua xử lý thường chứa nhiễu (noise) từ môi trường và các yếu tố sinh lý (ví dụ: nhịp tim, nháy mắt).
* Biểu đồ giúp bạn nhận diện các đặc điểm cơ bản của tín hiệu như: tần số dao động, cường độ tín hiệu, có nhiễu lớn hay không (ví dụ: biên độ đột biến bất thường).
* Kiểm tra xem tín hiệu có bị lỗi trong quá trình thu thập
* Tín hiệu EEG đã chia thành epoch:  
  
* Dữ liệu đã qua tiền xử lý (từ biến X), chia thành các epoch riêng lẻ.
* Mỗi đường trong biểu đồ biểu diễn tín hiệu EEG của điện cực đầu tiên (electrode 0) trong một epoch.
* Trục tung (V): Biên độ tín hiệu (đơn vị volt).
* Trục hoành: Số mẫu trong từng epoch.
* Tín hiệu EEG đã được cắt thành các đoạn nhỏ (epoch) với độ dài cố định, giúp bạn kiểm tra xem tín hiệu trong từng epoch có đặc điểm gì nổi bật (ví dụ: dao động alpha, beta).
* Các epoch này chính là dữ liệu đầu vào để huấn luyện mô hình học sâu, với mỗi epoch là một mẫu dữ liệu.

### 2.3 Mô hình (Model)

* Trong dự án này, ta sử dụng một mô hình học sâu được thiết kế để phân loại tín hiệu điện não đồ (EEG) sử dụng kiến trúc Transformer.
* Mô hình này được triển khai bằng cách sử dụng thư viện **PyTorch** và **PyTorch Lightning**, một framework hỗ trợ việc huấn luyện các mô hình học sâu.
* Tín hiệu EEG được tiền xử lý thông qua các lớp Conv1D (mạng tích chập một chiều) và sau đó được xử lý bởi một chuỗi các lớp Transformer nhằm trích xuất các đặc trưng không gian-thời gian.
* Ta sử dụng mô hình EEG Classification Model: Mô hình bao gồm ba phần chính:
* Lớp Conv1D: Tiến hành xử lý tín hiệu EEG, bao gồm các lớp tích chập với BatchNorm và ReLU để trích xuất các đặc trưng không gian. Lớp này giúp giảm độ nhiễu và cải thiện hiệu suất học.
* Lớp Transformer: Sau khi qua các lớp Conv1D, tín hiệu được chuyển đến một chuỗi các lớp Transformers Block, mỗi lớp bao gồm cơ chế self-attention để học các mối quan hệ dài hạn giữa các phần tử của tín hiệu EEG. Lớp Positional Encoding giúp mô hình hiểu được vị trí của các đặc trưng trong chuỗi thời gian.
* MLP (Multilayer Perceptron): Sau khi tín hiệu được xử lý qua Transformer, một MLP giúp tổng hợp các đặc trưng cuối cùng và đưa ra kết quả phân loại nhị phân.
* Ta sử dụng một lớp Avg Meter: Đây là lớp phụ trợ giúp theo dõi và tính toán các giá trị trung bình của mất mát (loss) và độ chính xác (accuracy) trong suốt quá trình huấn luyện và kiểm tra. Các giá trị này giúp giám sát sự cải thiện của mô hình qua các epoch.

### 2.4 Huấn luyện (Training)

* Ta khởi tạo các thông số:

**MAX\_EPOCH = 50**: Số lượng epoch tối đa là 50, tức là mô hình sẽ thực hiện tối đa 50 lần quét qua toàn bộ tập huấn luyện.

**BATCH\_SIZE = 10**: Kích thước mỗi batch trong huấn luyện là 10 mẫu, tức là trong mỗi lần cập nhật tham số, mô hình sẽ học từ 10 mẫu dữ liệu.

**LR = 5e-4**: Tốc độ học (learning rate) được thiết lập là 5×10−45 \times 10^{-4}5×10−4, ảnh hưởng đến tốc độ cập nhật trọng số của mô hình trong quá trình huấn luyện.

**CHECKPOINT\_DIR = os.getcwd()**: Đường dẫn thư mục nơi lưu các checkpoint của mô hình, trong trường hợp này là thư mục làm việc hiện tại.

**SEED = int(np.random.randint(2147483647))**: Tạo một giá trị seed ngẫu nhiên để đảm bảo tính tái lập cho quá trình huấn luyện. Điều này giúp kết quả huấn luyện có thể được tái tạo chính xác trong các lần chạy sau.

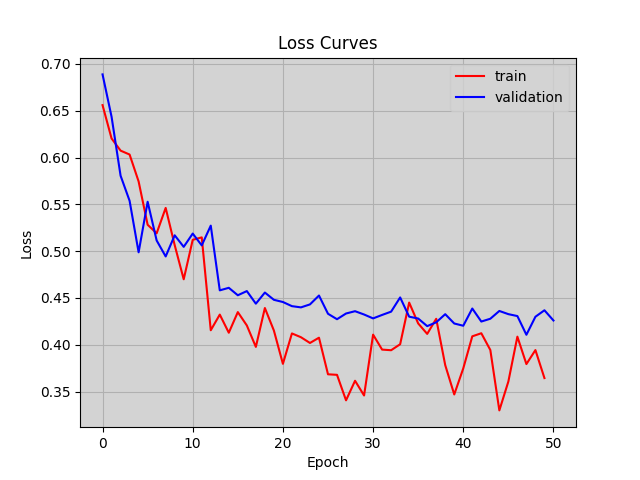
* Cấu hình Trainer:
* accelerator="auto": Tự động chọn phần cứng (GPU hoặc CPU) để huấn luyện.
* devices=1: Sử dụng 1 thiết bị cho quá trình huấn luyện (có thể là GPU nếu có).
* max\_epochs=MAX\_EPOCH: Mô hình sẽ huấn luyện tối đa 50 epoch.
* logger: Sử dụng cả tensorboardlogger và csvlogger để ghi lại thông tin huấn luyện.
* callbacks: Các callback như Learning Rate Monitor, ModelCheckpoint và EarlyStopping giúp giám sát quá trình huấn luyện và lưu lại mô hình tốt nhất.
* log\_every\_n\_steps=5: Ghi lại thông tin huấn luyện mỗi 5 bước (steps).
* trainer.fit(model): Bắt đầu quá trình huấn luyện mô hình với các cài đặt trên. Mô hình sẽ được huấn luyện trong các epoch đã định với sự giám sát của các callback và logger.
* Ta đã thiết lập một môi trường huấn luyện cho mô hình phân loại tín hiệu EEG, sử dụng các công cụ mạnh mẽ của PyTorch Lightning để theo dõi quá trình huấn luyện qua TensorBoard, lưu checkpoint của mô hình khi độ chính xác cải thiện và ngừng huấn luyện sớm khi không có sự tiến bộ. Các tham số như số epoch, kích thước batch và learning rate được chọn phù hợp để tối ưu hóa mô hình trong quá trình huấn luyện.

### 2.5 Kiểm tra (Testing)

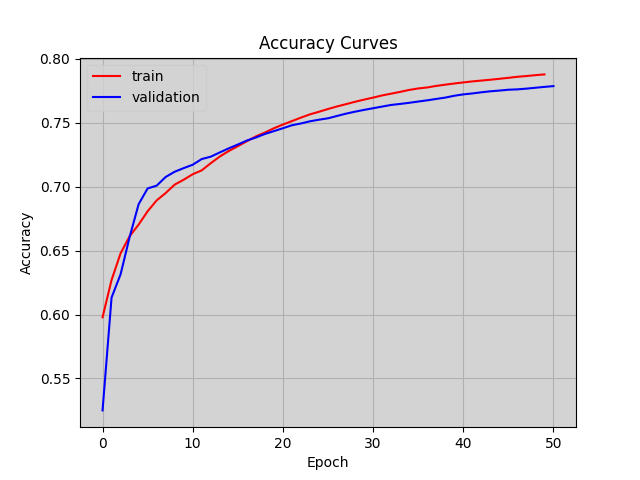
* Sử dụng lệnh trainer.test(ckpt\_path="best")
* Lệnh này thực hiện kiểm thử (test) mô hình đã huấn luyện với tập kiểm thử (test set).
* Đánh giá hiệu năng của mô hình trên tập dữ liệu kiểm thử (test set) để đo lường độ tổng quát hóa của mô hình
* Sử dụng lệnh os.rename(...)
* Việc đổi tên giúp dễ dàng nhận diện checkpoint này là mô hình tốt nhất trong các checkpoint đã lưu.

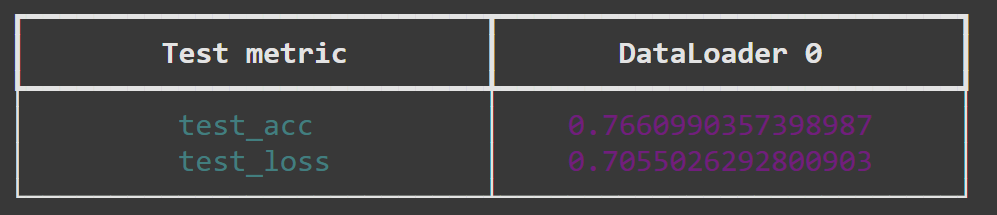
## Kết quả

* Kết quả training thu được:

**

* Đường loss trên tập huấn luyện (train loss) và validation đều giảm khi số epoch tăng, cho thấy mô hình đang học dần từ dữ liệu.
* **Giai đoạn đầu**: Loss giảm mạnh, điều này chỉ ra mô hình đã bắt đầu học được các đặc trưng cơ bản.
* **Từ khoảng epoch 15 trở đi**: Đường loss trên tập validation và train không giảm đáng kể nữa, và có những biến động nhỏ.
* **Hiện tượng**: Đường loss trên tập validation cao hơn đường train loss, cho thấy mô hình có thể đang **bắt đầu bị overfitting** (học quá tốt trên dữ liệu huấn luyện nhưng không tổng quát tốt trên dữ liệu kiểm tra).

**

* Độ chính xác trên cả hai tập (train và validation) tăng dần theo thời gian, đặc biệt trong **10 epoch đầu tiên**.
* **Train accuracy** cao hơn **validation accuracy** một chút, điều này phù hợp với hiện tượng overfitting nhẹ đã thấy trong biểu đồ loss.
* Độ chính xác trên tập validation dường như **bão hòa** ở khoảng **77-78%** sau epoch 40.
* **Xu hướng**: Mô hình đang học được và đạt được độ chính xác ổn định, tuy nhiên khoảng cách giữa train và validation accuracy cho thấy có thể cải thiện thêm.
* Kết quả testing thu được:
* Độ chính xác (test\_acc): Với độ chính xác khoảng 76.61%, mô hình đang hoạt động khá tốt trong việc phân loại tín hiệu EEG. Tuy nhiên, vẫn còn khoảng 23% kết quả sai, có thể cải thiện thêm nếu tối ưu hóa mô hình hoặc dữ liệu huấn luyện.
* Độ mất mát (test\_loss): Giá trị tổn thất 0.7055 cho thấy mô hình chưa đạt được hiệu quả tối đa. Mặc dù tổn thất đã giảm xuống, nhưng nó vẫn cần được cải thiện để mô hình có thể phân loại chính xác hơn.
* Các chỉ số này cho thấy mô hình đã học được phần lớn đặc trưng của dữ liệu, nhưng vẫn cần thêm sự tối ưu hóa, có thể thông qua các phương pháp như điều chỉnh hyperparameters, cải thiện dữ liệu huấn luyện, hoặc thay đổi kiến trúc mô hình.

## Kết luận

* Hiệu suất mô hình:
* Mô hình đã học tốt với dữ liệu EEG motor imagery, đạt được độ chính xác gần 78% trên tập validation.
* Có hiện tượng overfitting nhẹ vì train loss thấp hơn validation loss và train accuracy cao hơn validation accuracy.
* Hướng cải thiện:
* Regularization: Thêm kỹ thuật như Dropout hoặc L2 regularization để giảm overfitting.
* Early stopping: Dừng huấn luyện sớm khi loss trên tập validation không giảm đáng kể nữa.
* Dữ liệu: Thử nghiệm với các kỹ thuật augmentation dữ liệu EEG như sliding window hoặc tăng dữ liệu từ các tín hiệu EEG.
* Kiến trúc mô hình: Cải thiện kiến trúc mô hình, thử nghiệm các mô hình tiên tiến như CNN hoặc RNN dành cho dữ liệu thời gian.
* Kết luận: Mô hình đang hoạt động khá tốt có thể được sử dụng trong các ứng dụng thực tế như giao diện não máy tính (BCI), nơi mà việc phân loại tín hiệu EEG nhanh chóng và chính xác là rất quan trọng. Mặc dù kết quả hiện tại có thể cải thiện, mô hình này đã cho thấy tiềm năng lớn trong các ứng dụng như vậy.