Guided Project for 11-685

Exploring EEG-Image classification and EEG-Based Caption Retrieval

11-685: Introduction to Deep Learning (FALL 2025)

DUE: Oct. 31 (Midterm) / Dec. 10 (Final)

Start Here

- **Collaboration policy:**
  - You are expected to comply with the University Policy on Academic Integrity and Plagiarism.
  - You are allowed to talk and work with other students for homework assignments.
  - You can share ideas but not code, you must submit your own code. All submitted code will be compared against all code submitted this semester and in previous semesters using MOSS.
  - You are allowed to help your friends debug, however - you are not allowed to type code for your friend.
  - You are not allowed to look at your friends' code while typing your solution.
  - You are not allowed to copy and paste solutions off the internet.
  - Meeting regularly with your study group to work together is highly encouraged. You can even see from each other's solution what is effective, and what is ineffective. You can even "divide and conquer" to explore different strategies together before piecing together the most effective strategies. However, the actual code used to obtain the final submission must be entirely your own.
- **Overview:**
  - **Task 1**: Build EEG encoder for classification.
  - **Task 2**: Perform EEG-Caption Retrieval using your model and CLIP
- **Submission:**
  - **Report and Code**: should include your understanding of concepts, implementations, and evaluation results.
  - **Report Results**. You need to include the evaluation results for each part of this project, as specified in each individual subsection.

Checkpoints

- **Midterm Report, Due Oct. 31st at 11:59PM**: Baseline model for EEG classification and using pretrained CLIP model for Image-Caption retrieval.
- **Final Video, Due Dec. 5th at *6:00PM***.
- **Final Report, Due Dec. 10th at 11:59PM**.: Rest of the tasks.

Contents

1 Introduction

1.1 Problem Definition

When humans view images, their brains generate distinctive patterns of electrical activity that can be measured using electroencephalography (EEG). This project explores a fundamental question in neuroscience and machine learning: Can we decode what someone is seeing from their brain activity alone?

We address this question using a dataset of EEG recordings captured while participants viewed images from 20 different object categories (such as airplanes, bicycles, birds, boats, etc.). Each image also has an associated natural language caption describing its content. Rather than treating brain decoding as an isolated problem, we investigate whether EEG signals can be meaningfully aligned with multimodal representations-specifically, embedding spaces that jointly represent both visual and linguistic information.

1.1.1 Two Core Tasks

We formulate this brain decoding challenge as two distinct but related tasks that represent different levels of complexity: **classification** tests whether we can extract coarse categorical information from brain signals, while **retrieval** tests whether we can capture fine-grained semantic details that distinguish between similar images.

Task 1: Image Category Classification

- **Given:** EEG data recorded while a subject views an image.
- **Goal:** Predict which of the 20 object categories the image belongs to.

This is a supervised classification problem where the model must learn to map patterns in brain activity to discrete category labels. For example, if a subject is viewing an image of a bird, the model should predict *"bird"* from the EEG recording alone, without ever seeing the actual image.

Task 2: Caption Retrieval

- **Given:** EEG data recorded while a subject views an image.
- **Goal:** Retrieve the correct caption describing the image from a large pool of candidate captions.

This is an information retrieval problem that requires aligning neural representations with semantic text embeddings in a shared multimodal space. For example, if a subject is viewing an image with the caption *"a white and brown dog sitting on grass,"* the model should rank this caption higher than unrelated captions like *"a red airplane flying in the sky"* based solely on the EEG signals.

1.1.2 Approach Overview

Both tasks share a common foundation: we train neural networks to process raw EEG data and extract meaningful representations. For **classification**, these representations feed into a category predictor. For **retrieval**, we align EEG representations with text embeddings from CLIP (Contrastive Language-Image Pre-training), a large-scale vision-language model.

The key challenge across both tasks is handling the noisy, high-dimensional nature of EEG data while learning representations that generalize across substantial within-subject and between-subject variability (see Section 2 for details on sources of variation). Despite these challenges, models should extract the common neural patterns that emerge when different individuals view the same images.

1.2 Project Workflow

This project is organized around the two main tasks described above, each with specific subtasks designed to build understanding progressively before tackling the full brain decoding challenge.

1.2.1 Task 1: Image Category Classification

The first task establishes a baseline for EEG decoding by predicting which of 20 object categories an image belongs to based solely on brain activity.

- **Create a data structure for training:** Begin by familiarizing yourself with the dataset provided (see Section 2). Construct a data pipeline that integrates the processed EEG signals (`*.npy` files) with their corresponding image identifiers (`images.csv`), image categories, and textual captions (`captions.txt`). Ensure that the dataset is systematically partitioned into training, validation, and test subsets to support fair and reproducible evaluation.
- **Implement a baseline classifier:** Start with one of the baseline models described in the Baselines section 4.4 to understand baseline performance and data characteristics.
- **Design and train an EEG–Image Classifier:** Build a neural network architecture that jointly trains across all subjects using a **multi-head learning framework**. (see Section 3).
- **Evaluate and compare:** Test your model's classification accuracy and compare results against the basline you select to validate your implementation. See Section 4.5

1.2.2 Task 2: Caption Retrieval via Multimodal Alignment

The second task explores whether EEG signals can be aligned with language representations in a shared embedding space, enabling caption retrieval directly from brain activity. This task is divided into two parts.

Task 2A: Image-Caption Retrieval with CLIP

- **Understand CLIP's architecture:** Study how CLIP creates a shared embedding space for images and text through contrastive learning (see Section 5.2). Load a pre-trained CLIP model from Hugging Face and extract embeddings for both images and captions.
- **Implement image-to-caption retrieval and evaluate retrieval performance:** Pass images through CLIP's image encoder and captions through the text encoder. Return the top-K captions for each query image. Experiment with and report the evaluation metrics mentioned in Section 5.4.3

Task 2B: EEG-Caption Retrieval

- **Design EEG-to-CLIP projection:** Create a trainable projection head (linear or small MLP) that maps the EEG encoder's output into CLIP's text embedding space. Ensure all embeddings are normalized to unit length. See Section 5.5.1.
- **Experiment with CLIP fine-tuning strategies and perform joint training:** See Section 5.5.7.
- **Evaluate EEG-caption retrieval:** Repeat all analyses from Task 2A using EEG embeddings instead of image embeddings.

1.2.3 Experimental Considerations

Throughout both tasks, the following considerations should guide the experiments:

- An example EEG encoder architecture will be provided (see Section 3). You may use this design as-is or propose your own variant. Grading will emphasize the quality of your implementation.
- Evaluate generalization across subjects and sessions.
- Reflect critically on results: What works? What are the limitations? What does this tell us about EEG-language alignment?

1.3 Homework Grading

The homework grading will be based on your report and code implementation:

- 40% - Design and train an EEG encoder for image category classification
- 20% - Practicing with CLIP for Image–Caption retrieval and evaluation
- 30% - Experiment with advanced techniques for CLIP fine-tuning. Use them for aligning EEG and text embeddings within CLIP's joint space for caption retrieval
- 10% - Evaluation of captions retrieved from EEG embeddings

2 Dataset Description

This project utilizes a subset of the Multi-Subject and Multi-Session electroencephalography (EEG) dataset introduced by Xue et al [1]. The dataset contains EEG recordings captured while subjects viewed visual stimuli from PASCAL VOC and ImageNet databases. The goal is to investigate whether brain activity patterns can be meaningfully aligned with semantic representations of images and their textual descriptions.

2.1 Electroencephalography (EEG): An Overview

EEG is a non-invasive approach to record brain activity through electrodes placed on the scalp. It measures voltage fluctuations reflecting the synchronous firing of cortical neurons. Compared to fMRI or MEG, EEG has the advantages of low cost, portability, and millisecond temporal resolution, which makes it a popular tool for brain–computer interface (BCI) research and cognitive neuroscience. However, EEG is susceptible to artifacts such as eye blinks or muscle activity, exhibits strong subject/session variability, and suffers from low spatial resolution.

Data from EEG are typically structured as multichannel time series: for C electrodes and T time samples, each trial produces a $C \times T$ matrix. Data preprocessing often includes downsampling, bandpass filtering, and artifact rejection.

A critical challenge in EEG-based decoding is the substantial variability both *within* and *between* subjects. Within a single subject, recordings vary significantly across sessions due to factors such as electrode cap repositioning, fluctuations in attention and fatigue, and day-to-day physiological differences. Between subjects, individual differences in brain anatomy, skull thickness, and neural response patterns create even larger variations. Any successful decoding model must learn representations robust to these sources of noise while still extracting the common neural signatures that emerge when different individuals view the same image.

2.2 Dataset Components

The Dataset can be found on PSC. To get access to the dataset directory, please fill out this form. Note that all members of your project group must fill it out to get access to the data (due to copyright clauses).

The dataset comprises three primary components that enable multimodal alignment between brain signals, images, and language:

2.2.1 Visual Stimuli

A collection of 10,000 images sourced from PASCAL VOC and ImageNet datasets, spanning 20 object categories with 500 images per category. Images are stored as JPEG / JPG files in the `images/` directory, each identified by a unique filename (e.g., `000002.jpg`).

2.2.2 Caption Annotations

Natural language descriptions for each image, stored in `captions.txt`. These captions serve as ground-truth text for both retrieval and generation tasks.

2.2.3 EEG Recordings

Brain activity recorded from 13 subjects during visual stimulus presentation. Each recording captures neural responses across 122 scalp electrodes at a sampling rate of 1000 Hz. The preprocessed EEG data are stored as NumPy arrays in `.npy` files.

```
ds005589/
■■■ sub-XX/ (Subject directories, 13 total)
■■■ ses-YY/ (Session directories, 5 per subject)
■■■ ■■■ *_task-lowSpeed_run-ZZ_1000Hz.npy (EEG data, low-speed)
■■■ ■■■ *_task-lowSpeed_run-ZZ_1000Hz.json (Metadata)
■■■ ■■■ *_task-lowSpeed_run-ZZ_image.csv (Image trial info)
■■■ ■■■ ... (Other paradigms, e.g., RSVP or high-speed)
images/
■■■ XXXXXX.jpg (Visual stimuli,~10,000 images)
■■■ captions.txt (Caption annotations)
```

Figure 1: Dataset directory structure showing the hierarchical organization of EEG recordings by subject, session, and run

2.3 Data Organization and Experimental Design

The dataset follows a hierarchical structure with 13 participants (`sub-02`, `sub-03`, `sub-05`, `sub-09`, `sub-14`, `sub-15`, `sub-17`, `sub-19`, `sub-20`, `sub-23`, `sub-24`, `sub-28`, `sub-29`). Each subject completed 5 recording sessions on different days (`ses-01` through `ses-05`).

Each session contains data from two experimental paradigms:

1. **Low-speed task** (`task-lowSpeed`): Images presented for 1 second, followed by a multiple-choice classification task to maintain subject engagement. Each session includes 4 runs with 100 trials per run (400 trials total per session).
2. **Rapid Serial Visual Presentation (RSVP)** (`task-rsvp`): Images presented for 0.1 seconds in rapid succession. Each session includes 4 runs with 2,000 trials per run (8,000 trials total per session).

For this project, we focus exclusively on the **low-speed paradigm**. The longer presentation duration (1 second) provides non-overlapping EEG signals with stronger trial-level alignment, making it more suitable for training models that map EEG responses into multimodal embedding spaces.

2.3.1 Hierarchical Data Structure

The dataset follows this hierarchy:

- **Subjects**: 13 participants
- **Sessions**: 5 sessions per subject (recorded on different days)
- **Runs**: 4 runs per session
- **Trials**: 100 trials per run (each trial = one image presentation)

This structure means each subject contributes 2,000 trials ($5 \times 4 \times 100$), yielding 26,000 total trials across all subjects.

Importantly, this multi-session, multi-subject design introduces substantial variability: the EEG cap is removed and repositioned between sessions, electrode impedances change, and subjects experience different cognitive states across recording days. Combined with inherent inter-subject anatomical and neural differences, this creates a challenging learning problem where models must discover invariant features across highly variable signals.

2.4 EEG Preprocessing

The EEG data have been preprocessed following the procedures outlined in the original study using the MNE Python package [2]:

- **Bandpass filtering (0.1–100 Hz)**: Removes slow drifts and high-frequency muscle artifacts while retaining physiologically relevant EEG rhythms (delta, theta, alpha, beta, gamma)
- **Notch filtering at 50 Hz**: Suppresses electrical line noise
- **Epoching from 0–500 ms post-stimulus**: Segments continuous EEG into trials aligned with visual stimulus onset, capturing early and mid-latency visual processing

2.5 EEG Data Dimensions

After preprocessing, each run produces a data matrix of dimensions ($100 \times 122 \times 500$), representing 100 trials across 122 channels and 500 time points. For a complete session (4 runs), this yields ($400 \times 122 \times 500$). When aggregating across all subjects and sessions, the final input data has dimensions:

$$(n_{\text{subjects}} \times n_{\text{sessions}} \times 4) \times 122 \times 500 \tag{1}$$

where the first dimension represents the total number of trials across all experimental recordings (26,000 trials for 13 subjects), 122 is the number of channels, and 500 is the number of time points (500 ms at 1000 Hz sampling rate).

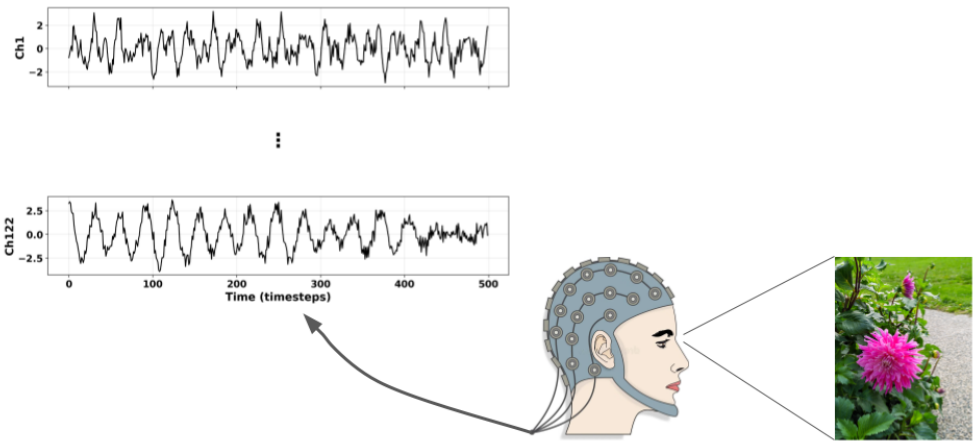

Figure 2: EEG signals from a single trial showing activity across 122 channels during one image presentation (0–500 ms post-stimulus)

2.6 File Structure

The dataset follows Brain Imaging Data Structure (BIDS) format. Each experimental session directory contains:

- `sub-XX_ses-YY_task-lowSpeed_run-ZZ_1000Hz.npy`: EEG data array [trials × channels × timepoints]
- `sub-XX_ses-YY_task-lowSpeed_run-ZZ_1000Hz.json`: Metadata including channel names, sampling rate, and experimental parameters
- `sub-XX_ses-YY_task-lowSpeed_run-ZZ_image.csv`: Trial-to-image correspondence mapping

2.7 Data Loading Procedure

To construct multimodal training pairs *(EEG trial, image name, image semantic category, caption)*, the following steps are required:

1. **Parse metadata:** Read the run-specific CSV file:

   `{SUB}_{SES}_task-lowSpeed_run-{RID}_image.csv`

   to obtain the image identifiers for each trial, using the `FilePath` column and extracting the base image name.

2. **Load EEG trials:** Load the corresponding EEG data from the preprocessed `.npy` file (`{SUB}_{SES}_task-lowSpeed_run-{RID}_1000Hz.npy`) using `numpy.load()`, which returns an array of shape $[N, 122, T]$, where N is the number of trials, 122 is the number of electrodes, and T is the time dimension.

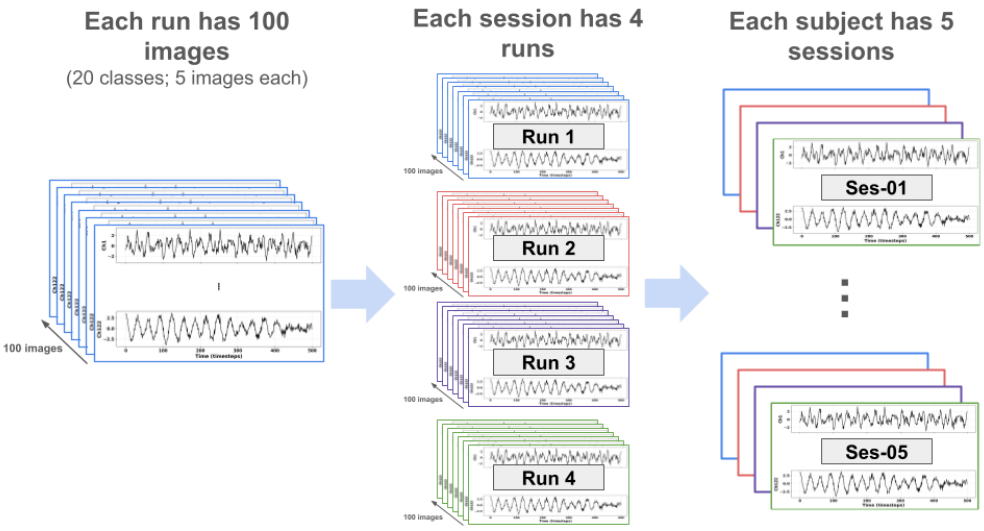

Figure 3: Combined EEG data aggregated across multiple trials, showing the overall data structure

3. **Merge with captions:** Join the image identifiers with the captions file (`captions.txt`) on the `image_name` column to retrieve the associated caption and semantic category for each trial.

4. **Resolve image paths:** Match the image identifiers to their corresponding files in the `All_images/` directory, supporting `.jpg`, `.jpeg`, and `.png` extensions.

5. **Verify correspondence:** Confirm that the number of trials in the CSV file matches the number of EEG arrays in the `.npy` file to ensure one-to-one alignment between modalities.

Train/validation/test split. For each complete subject in the dataset, the data should be split by session: three sessions for training, one session for validation, and one session for testing. This session-based split ensures temporal separation and prevents data leakage across splits.

This procedure establishes the correspondence necessary for training both classification models (Section 4) and retrieval models (Section 5.5).

3 Architecture Overview

Our approach to EEG decoding employs a unified multi-stage neural architecture that trains across all subjects concurrently through a **multi-head learning framework**. Rather than building separate models for each subject, we use a shared architecture with subject-specific output heads. This design allows the model to learn general-purpose EEG representations that capture visual semantic information, while still accounting for subject-level variabil-

ity through specialized heads. By training all subjects together, the shared backbone can discover more robust and generalizable features than training on any single subject alone.

The architecture is designed to handle the challenging characteristics of brain signals: high dimensionality, temporal dependencies, and substantial inter-subject variability. It consists of three main components that process data sequentially: a convolutional feature extractor, a shared transformer backbone, and subject-specific prediction heads.

Importantly, we employ the same base architecture for both the classification and retrieval tasks. The training strategy allows the outputs to support either classification or caption retrieval.

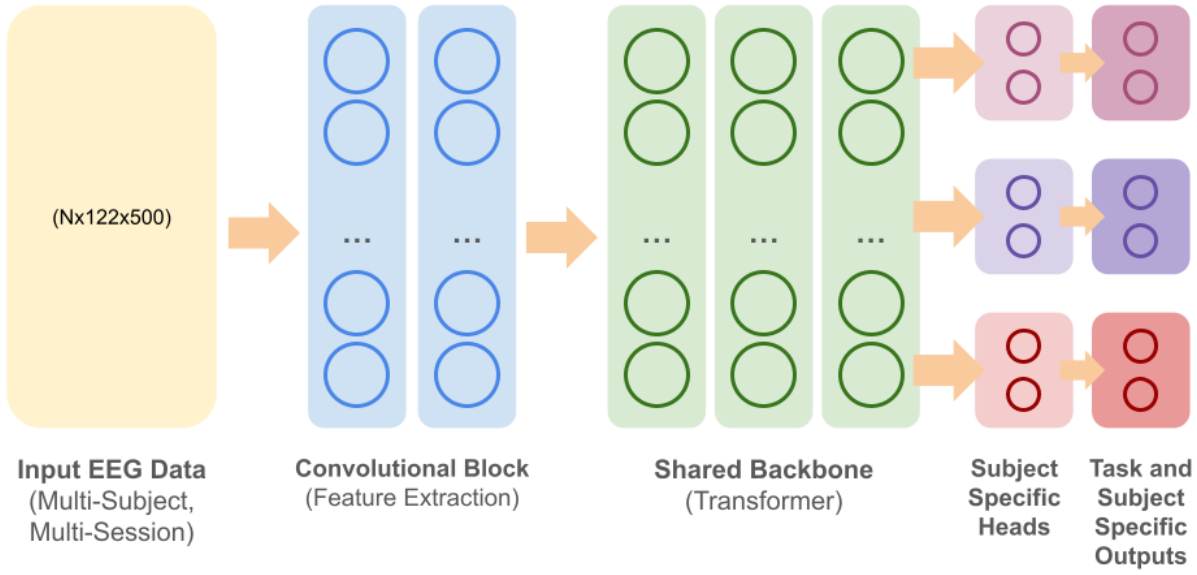

Figure 4: Main architecture description, featuring the feature extractor, backbone, and projection head components

3.1 Stage 1: Convolutional Feature Extractor

We begin by processing the EEG input tensor of shape 122×500 (channels $\times$ timepoints) with a CNN based feature extraction module. This module employs one-dimensional convolutions that operate along the temporal axis of each channel, treating each of the 122 electrodes as an independent 1D time series of length 500.

For each channel, the convolutional layer applies $n_{\text{out\_channels}}$ filters that slide across the temporal dimension, thereby learning localized temporal features such as transient oscillatory activity or event-related modulations. The output of this step has dimensions

$$122 \times n_{\text{out\_channels}} \times n_{\text{out\_times}},$$

where $n_{\text{out\_times}}$ reflects any temporal reduction from striding or pooling operations.

To obtain a compact representation, temporal pooling (typically mean pooling across the time axis) collapses the temporal dimension, producing a fixed-length feature matrix of shape

$$122 \times n_{\text{out\_channels}}.$$

In practice, this representation is often further projected onto a target embedding dimension, for example 122×64, such that each electrode is summarized by a 64-dimensional vector encoding its temporal dynamics.

This transformation performs dimensionality reduction by removing noise and irrelevant fluctuations while extracting task-relevant temporal signatures. Importantly, the spatial topography of the electrode array is preserved: each electrode retains its identity and position, but its raw 500-sample waveform is replaced with a compact learned embedding. This spatially-preserved, temporally-compressed representation forms a structured input for the subsequent transformer backbone, which can then model spatial relationships and dependencies across electrodes.

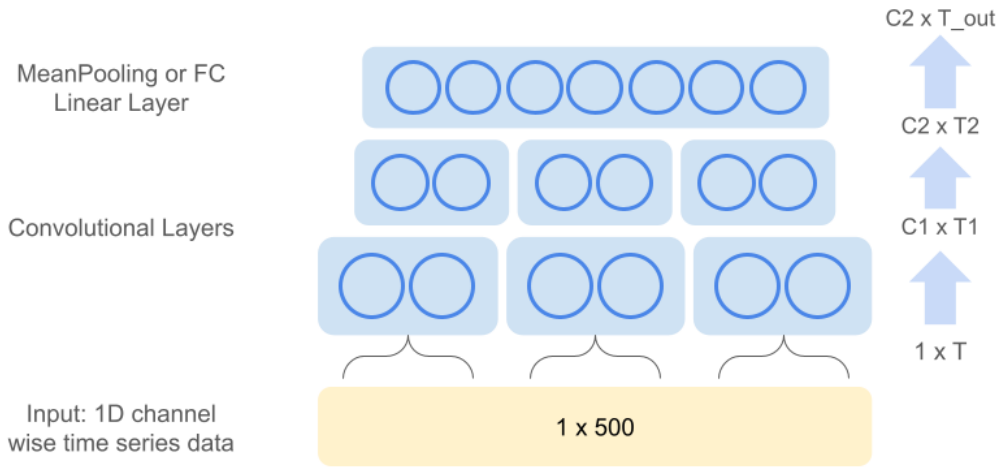

Figure 5: CNN Feature Extraction Module. The figure shows the 1D convolution and pooling pipeline for one representative channel. This channel-wise operation is applied across all 122 EEG channels to extract temporal features while preserving spatial topology.

3.2 Stage 2: Transformer Based Backbone

The extracted CNN features feed into a transformer-based backbone that processes sequences and learns relationships across time and trials. Critically, this backbone is **shared across all subjects**. During training, it receives data from all 13 subjects and learns a common feature space that captures universal aspects of visual processing (the neural patterns that

emerge when any human views a particular category or image). By learning from multiple subjects jointly, the model may discover robust, generalizable representations rather than overfitting to individual-specific data.

3.3 Stage 3: Subject-Specific Prediction Heads

After the shared backbone, the architecture splits into N separate prediction heads, one for each subject. Each head can be implemented as additional transformer layers, LSTM networks, or simple feed-forward classifiers, depending on the task requirements.

This multi-head design addresses the substantial between-subject variability discussed in Section 2. While the shared backbone learns common patterns, the subject-specific heads learn individual calibration factors: how each person's unique brain anatomy, electrode positioning, and neural response characteristics map to the task labels.

The output layer of each head differs by task:

- **Classification**: Each head outputs logits over the 20 object categories, trained with cross-entropy loss
- **Retrieval**: Each head produces embedding vectors aligned with CLIP's text embedding space, trained with contrastive loss

3.4 Training Strategy: Selective Backpropagation

The training procedure carefully balances learning shared and subject-specific representations:

- **Forward pass:** A batch of trials passes through the CNN and shared backbone. The backbone output then feeds into all 13 subject-specific heads simultaneously.
- **Loss computation:** Each head computes loss only for trials from its corresponding subject. For example, if a batch contains 32 trials from subject-02, only the subject-02 head's loss is computed for those trials.
- **Backward pass with masking:**
  - Gradients from each subject's head flow back through the shared backbone
  - Head parameters are updated only for the subject(s) present in the current batch
  - The CNN and backbone receive gradients from all subjects over the course of training

This masking strategy ensures that:

- Subject-specific heads specialize to individual patterns without interference from other subjects' data
- The shared components (CNN and backbone) benefit from the full diversity of the dataset
- The model avoids negative transfer where one subject's training degrades performance on others

3.5 Architectural Flexibility

The modular design allows experimentation with different components:

- **CNN architecture:** Deep ConvNets, temporal convolutions
- **Backbone:** Transformer encoders or hybrid architectures
- **Head design:** Simple linear layers, additional transformer blocks, or recurrent layers

4 Task 1: Classification

4.1 Objective

The classification task requires predicting which of the 20 object categories an image belongs to based solely on the EEG signals recorded while a subject viewed that image. This is a supervised learning problem where the model must learn to map patterns in brain activity to discrete category labels.

4.2 Architecture output for Classification

For this task, you will create task and subject specific output heads. See Section 3 for details on the network architecture.

4.3 Classification Loss Function

The model is trained using **cross-entropy loss**. For a single trial with true label c and predicted logits $\mathbf{z}$, the softmax function converts logits to probabilities:

$$p_i = \frac{\exp(z_i)}{\sum_{j=1}^{\text{num\_classes}} \exp(z_j)}$$

The cross-entropy loss is then:

$$\mathcal{L}_{\text{CE}}(\mathbf{z}, c) = -\log(p_c) = -z_c + \log\left(\sum_{j=1}^{\text{num\_classes}} \exp(z_j)\right)$$

This loss is minimized when the model assigns high probability to the correct class.

4.4 Baseline Models for EEG Decoding

To establish performance benchmarks for our EEG-to-image classification tasks, we consider several neural architectures that have been validated on EEG visual recognition tasks. These baselines range from simple fully connected networks to specialized convolutional architectures designed for EEG signals. You can choose from one of these baselines, or come up with your own. Make sure to justify your choice in the writeup!

Validation baselines from the original dataset. Wang *et al.* [1] validated their dataset using a fully connected neural network classifier for category-level visual decoding from EEG signals. Their results demonstrated the feasibility of extracting visual information from EEG activity. In the within-session classification setting (same subject, same session), the model achieved an average accuracy of approximately 10%, exceeding the 5% chance level expected from random guessing. However, performance varied across participants, with some reaching accuracies up to 16%, while others were closer to chance level. When evaluated in cross-session and cross-subject settings, the classification accuracy further declined, highlighting the challenges of generalizing EEG-based visual decoding across recording sessions and individuals.

Despite the reported variability, these findings suggest that object category information is present in EEG signals, indicating that the dataset is suitable for EEG decoding tasks. You can choose any of the implemented models in the original paper Wang *et al.* [1] to serve as a baseline.

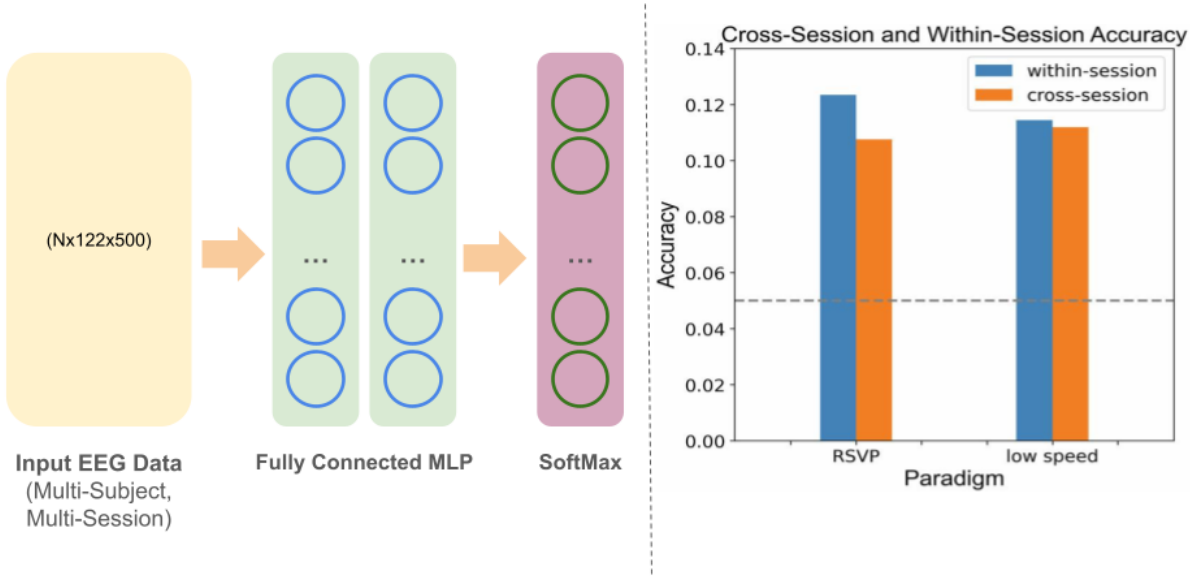

Figure 6: Baseline Model used used by Wang et al [1]

4.5 Evaluation

The primary metric is classification accuracy:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

In addition, you are required to compute the **confusion matrix** across categories and report per-subject accuracy.

5 Task 2: Caption Retrieval

5.1 Objective

The caption retrieval task explores whether EEG signals can be mapped into a shared embedding space with natural language to enable text retrieval from brain activity. Unlike classification, which assigns discrete labels, retrieval requires learning continuous embeddings where semantically similar concepts lie close together. This task is structured in two parts:

Task 2A: Image-Caption Retrieval with CLIP As a warm-up, you will work with CLIP (**C**ontrastive **L**anguage–**I**mage **P**retraining), a model pretrained on 400 million image-text pairs to create a shared embedding space. You will use CLIP to perform image-to-caption retrieval, gaining familiarity with multimodal embeddings, similarity-based retrieval, and various evaluation metrics (Recall@K, BLEU, BERTScore). This establishes a baseline for understanding how well retrieval works when using actual images.

Task 2B: EEG-Caption Retrieval You will train an EEG encoder to map brain signals into CLIP's text embedding space, enabling EEG-to-caption retrieval. You will experiment with different training strategies (frozen vs. fine-tuned CLIP), loss functions (knowledge distillation, contrastive learning), and parameter-efficient fine-tuning methods.

5.2 Background: CLIP (Contrastive Language–Image Pretraining)

$$\begin{array}{rcccc} \text{Image} & \rightarrow & \text{Image Encoder} & \rightarrow & z_{\text{img}} \\ \text{Text} & \rightarrow & \text{Text Encoder} & \rightarrow & z_{\text{text}} \end{array} \rightarrow \text{Similarity}$$

CLIP was introduced by Radford et al. [3]. The key idea behind CLIP is to learn a *shared embedding space* where images and their corresponding text descriptions are mapped to nearby points, while unrelated image-text pairs are pushed apart. This joint representation enables cross-modal tasks like retrieving captions for images or finding images that match text queries (see Figure 7).

CLIP consists of two encoders that operate independently:

- An **image encoder** f_θ (typically a Vision Transformer or ResNet) that maps images to vectors:
$$\mathbf{img} = f_\theta(I) \in \mathbb{R}^d.$$
- A **text encoder** g_ϕ (typically a Transformer) that maps captions to vectors:
$$\mathbf{txt} = g_\phi(T) \in \mathbb{R}^d.$$

These components then undergo the following steps:

Step 1: Encode and normalize. The encoders produce vectors **img** and **txt** for images and text separately, which are normalized to unit length:

$$\mathbf{img}_z = \frac{\mathbf{img}}{\|\mathbf{img}\|_2}, \qquad \mathbf{txt}_z = \frac{\mathbf{txt}}{\|\mathbf{txt}\|_2}.$$

Step 2: Cosine similarity. After normalization, the similarity between one image I and one caption T is computed using the dot product of their normalized embeddings:

$$\text{sim}(I, T) = \mathbf{img}_z^\top \mathbf{txt}_z.$$

Higher similarity scores indicate that the image and text are semantically related, while lower scores indicate they are unrelated.

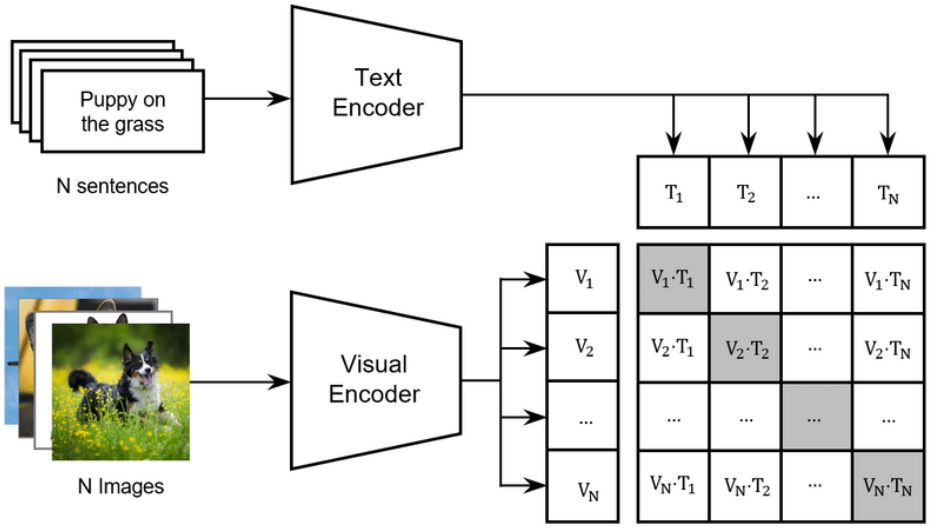

Figure 7: Illustration of CLIP training: paired images and texts are encoded into a shared embedding space, and then similarities are computed across all $N \times N$ pairs in the batch. Diagonal elements (matched pairs) are contrasted against off-diagonal elements (negatives). Reproduced from Song et al. [4], *"CLIP Models are Few-Shot Learners: Empirical Studies on VQA and Visual Entailment,"* licensed under CC BY-NC-ND 4.0.

Step 3: Batch similarity matrix. CLIP is trained using *contrastive learning*, which teaches the model to distinguish correct image–text pairs from incorrect ones. During training, CLIP processes a batch of N image–caption pairs. For each batch, it computes an $N \times N$ similarity matrix S where:

$$S_{ij} = \mathbf{img}_{z,i}^\top \mathbf{txt}_{z,j}.$$

The diagonal elements S_{ii} represent correct matches (image i with its true caption), while off-diagonal elements represent mismatches. To control the sharpness of the similarity distribution, CLIP scales the matrix by a temperature parameter τ:

$$Z_{ij} = \frac{S_{ij}}{\tau}.$$

Step 4: Softmax probabilities. For each image i, CLIP computes a probability distribution over all captions in the batch using softmax:

$$p_{i\to j} = \frac{\exp(Z_{ij})}{\sum_{k=1}^{N} \exp(Z_{ik})}.$$

Similarly, for each caption j, it computes a distribution over all images:

$$p_{j\to i} = \frac{\exp(Z_{ij})}{\sum_{k=1}^{N} \exp(Z_{kj})}.$$

Step 5: Contrastive loss. The training objective is to maximize the probability of correct matches. This is achieved using cross-entropy loss in both directions (image-to-text and text-to-image), then averaging:

$$\mathcal{L} = \tfrac{1}{2}\Big(\mathcal{L}_{i\to t} + \mathcal{L}_{t\to i}\Big),$$

where

$$\mathcal{L}_{i\to t} = -\frac{1}{N}\sum_{i=1}^{N} \log p_{i\to i} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(Z_{ii})}{\sum_{j=1}^{N} \exp(Z_{ij})},$$

and

$$\mathcal{L}_{t\to i} = -\frac{1}{N}\sum_{j=1}^{N} \log p_{j\to j} = -\frac{1}{N}\sum_{j=1}^{N} \log \frac{\exp(Z_{jj})}{\sum_{i=1}^{N} \exp(Z_{ij})}.$$

This bidirectional loss ensures that the model learns symmetric similarity: if an image matches a caption, the caption should also match that image.

The Role of Temperature The temperature parameter τ controls the sharpness of the probability distribution. Consider four vectors with raw similarity scores $[2.0, 1.0, 0.5, 0.1]$. Applying softmax with different temperatures produces:

```
tau = 0.1 → [0.9999, 0.0000, 0.0000, 0.0000]  (sharp, confident)
tau = 1.0 → [0.5761, 0.2120, 0.1282, 0.0837]  (moderate)
tau = 2.0 → [0.3790, 0.2506, 0.1971, 0.1733]  (smooth, uncertain)
```

A small τ creates sharp distributions that strongly separate positives from negatives, while a large τ creates smoother distributions that spread probability more evenly. During CLIP training, τ is typically learned as a parameter.

5.3 Applications of CLIP

CLIP has multiple applications including image-caption retrieval, zero-shot classification, multimodal search, guiding image generation [5] (Stable Diffusion), and cross-domain tasks such as in robotics [6] (video-text), or in medical applications [7] (pathology / radiology image-report alignment).

The shared multimodal embedding space learned by CLIP lies at the core of these applications. Within this space, matched pairs are trained to lie close together, whereas pushing apart mismatched pairs, enabling cross-modal retrieval. This joint representation highlights the ability of CLIP to generalize across tasks. Downstream problems can be reframed as embedding comparison in the shared space.

5.4 Task 2A: Image-Caption Retrieval with CLIP

Before working with EEG signals, you will first familiarize yourself with CLIP by performing image-to-caption retrieval. This warm-up exercise will help you understand how retrieval works in CLIP's embedding space and establish baseline performance that you can later compare against EEG-based retrieval.

5.4.1 Setup and Implementation

Loading pretrained CLIP. You will load a pretrained CLIP model from the Hugging Face Transformers library.

Computing embeddings. For each image in your dataset, you will pass the image through CLIP's pretrained image encoder to produce an image embedding:

$$\mathbf{img} = f_\theta(I) \in \mathbb{R}^d.$$

Similarly, for each caption, you will pass the text through CLIP's pretrained text encoder to produce a text embedding:

$$\mathbf{txt} = g_\phi(T) \in \mathbb{R}^d.$$

Performing retrieval via cosine similarity. To retrieve captions for a given image, you will:

1. Compute the cosine similarity between the normalized image embedding and all caption embeddings:
$$s_j = \mathbf{img}_z^\top \mathbf{txt}_{z,j}, \quad j = 1, \ldots, M,$$
where M is the total number of available captions.
2. Rank the captions by similarity score in descending order.
3. Return the top-K captions as the retrieval results.

5.4.2 Evaluation Metrics for Caption Retrieval

To evaluate retrieval performance, you will experiment with different semantic metrics, and then select best ones to use in the EEG-caption retrieval task.

Retrieval metrics: Recall@K. Recall@K measures whether the correct caption appears within the top-K retrieved results. For a query image i with ground-truth caption g_i and retrieved caption set $\{r_{i1}, \ldots, r_{iK}\}$:

$$\text{Recall@K} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[g_i \in \{r_{i1}, \ldots, r_{iK}\}].$$

You will compute Recall@K in two ways:

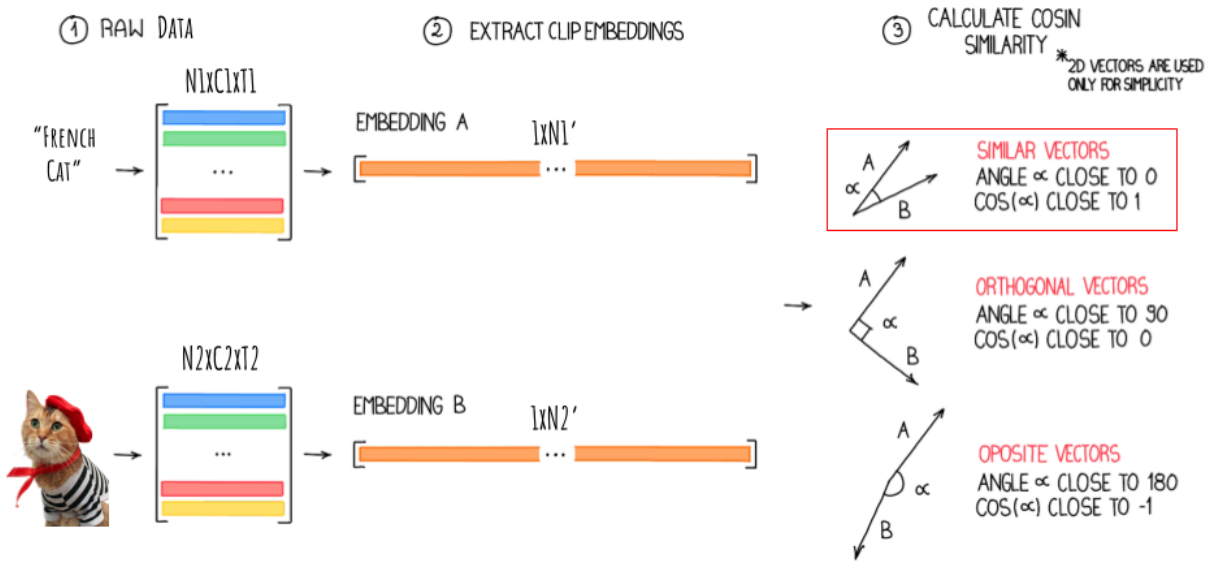

Figure 8: Visual Description of Cosine Similarity. Image adapted from Roboflow

- **Instance-level Recall@K:** retrieval is correct only if the exact ground-truth caption is retrieved within the top-K.
- **Class-aware Recall@K:** retrieval is correct if any caption from the same semantic class as the ground-truth is retrieved within the top-K. This is more lenient and reflects the fact that multiple captions may describe similar visual content.

Example: Query = train image. Retrieved top-5 captions = [cat, airplane, dog, train, house]. If ground truth = "train," then R@1 = 0 (not first), R@5 = 1 (within top-5).

Semantic metrics: BERTScore and CLIPScore. Semantic metrics capture meaning-level similarity rather than exact word matches.

BERTScore [8] computes token embeddings using a pretrained Transformer (e.g., RoBERTa). For captions C and R:

$$P = \frac{1}{|C|}\sum_{x\in C}\max_{y\in R}\cos(e(x),e(y)), \quad R = \frac{1}{|R|}\sum_{y\in R}\max_{x\in C}\cos(e(x),e(y)), \quad F1 = \frac{2PR}{P+R}.$$

CLIPScore [8] computes similarity between image and text embeddings.

5.4.3 Analysis Requirements

You are required to conduct the following analyses to understand CLIP's retrieval behavior:

- Report instance-level and class-aware **Recall@1, Recall@3, and Recall@5.** Explain what these numbers tell you about CLIP's ability to retrieve exact matches versus semantically similar captions.
- Report the **BERTScore** for the test captions. Note the success rate of high semantic similarity (BERTScore F1 > 0.7).
- Compare between **CLIPScore** distribution of matched and mismatched image-caption pairs.
- Mean Average Precision (MAP):
  - *Caption-level MAP (Overall)*: average precision across all queries considering only the exact ground-truth caption as relevant.
  - *Class-aware MAP*: computes MAP when any caption from the same semantic class is treated as relevant.
  - *Per-class MAP*: compute MAP separately for each semantic class.

5.5 Task 2B: EEG-Caption Retrieval

Our goal is to train an EEG encoder to align its outputs with CLIP's text embeddings for image captions. This allows us to perform *EEG → Caption Retrieval*: from the shared embedding space, given an EEG trial recorded while viewing an image.

$$\begin{array}{llllll} \text{EEG} & \rightarrow & \text{EEG Encoder} & \rightarrow & \text{Projection} \rightarrow z_{\text{eeg}} & \rightarrow \\ \text{Text} & \rightarrow & \text{Text Encoder} & \rightarrow & z_{\text{text}} & \rightarrow \end{array} \quad \text{Similarity}$$

5.5.1 Architecture Overview

You will use the same EEG encoder architecture from Task 1 (see Section 3). For each subject, you will create a new trainable projection head that maps the EEG encoder's output into CLIP's d-dimensional text embedding space. This projection head can be either a linear layer or a small multi-layer perceptron (MLP) with one to two hidden layers.

After projection, normalize the embedding to unit length, similar to the original CLIP model:

$$\mathbf{z}_{\text{eeg},z} = \frac{\mathbf{z}_{\text{eeg}}}{\|\mathbf{z}_{\text{eeg}}\|_2}.$$

Following this, your projection head is ready for training.

5.5.2 Loss Functions

We will consider complementary losses that (i) *align* EEG embeddings with CLIP's caption space, (ii) *shape* the embedding geometry via contrastive learning, and (optionally) (iii) *regularize* with category supervision.

5.5.3 Knowledge Distillation (Teacher-Student)

You are encouraged to experiment with different knowledge distillation (KD) strategies, as surveyed in [9] and applied to CLIP in [10]. Broadly, KD methods can be grouped into three categories:

- **Feature-based:** the student matches hidden activations of the teacher. For example: aligning intermediate EEG encoder features to CLIP's image encoder features. We will not use this in our project.
- **Similarity-based:** the student reproduces teacher similarities or direct embedding alignments. In our setup, the EEG encoder + projection head is trained to align directly with the caption embedding of its ground-truth:

  $$\mathcal{L}_{\cos} = 1 - \mathbf{eeg}_z^\top \mathbf{txt}_z,$$

  This stabilizes training and may anchor EEG embeddings near their ground-truth caption embeddings.
- **Logit-based:** the student mimics the teacher's probability distribution over candidates. Since each EEG trial corresponds to an image I, and CLIP already knows how to compare images and captions, so CLIP's image encoder, paired with its text encoder, can produce a teacher distribution over captions. $\{T_j\}_{j=1}^M$:

  $$q_j = \frac{\exp(\mathrm{sim}(\mathbf{img}_z, \mathbf{txt}_{z,j})/\tau_t)}{\sum_{k=1}^M \exp(\mathrm{sim}(\mathbf{img}_z, \mathbf{txt}_{z,k})/\tau_t)}, \quad p_j = \frac{\exp(\mathrm{sim}(\mathbf{eeg}_z, \mathbf{txt}_{z,j})/\tau_s)}{\sum_{k=1}^M \exp(\mathrm{sim}(\mathbf{eeg}_z, \mathbf{txt}_{z,k})/\tau_s)}.$$

  We will need the student distribution p to match the teacher distribution q, so the loss can be KL divergence:

  $$\mathrm{KL}(q \,\|\, p) \;=\; \sum_{j=1}^M q_j \log \frac{q_j}{p_j}.$$

  This teaches the EEG encoder to reproduce CLIP's *ranking* over many captions, which can be more robust to noise than enforcing a single hard target.

5.5.4 Contrastive Alignment

Naïve InfoNCE (instance positives, all others negatives). For a batch $\{(\mathbf{eeg}_{z,i}, \mathbf{txt}_{z,i})\}_{i=1}^N$ with logits $Z_{ij} = \mathbf{eeg}_{z,i}^\top \mathbf{txt}_{z,j}/\tau$, the standard InfoNCE loss is:

$$\mathcal{L}_{\mathrm{NCE}}^{\mathrm{E}\to\mathrm{T}} = -\frac{1}{N}\sum_{i=1}^N \log \frac{\exp(Z_{ii})}{\sum_{j=1}^N \exp(Z_{ij})}.$$

Problem. The naïve InfoNCE formulation assumes each EEG-caption pair (i, i) is the only positive, and *all other captions in the batch are negatives*. However, EEG data may not be highly discriminative, especially given the relatively low within-session classification accuracy reported in the original paper [1]. This means that many other captions $j \neq i$ "may" evoke neural responses that look similar at the EEG level (e.g., two "airplane" captions). If this might be the case, treating these as strict negatives forces the model to push apart semantically valid pairs, introducing *false negatives*.

Option: Debiased / soft negatives [11]. Instead of treating all non-matching captions as equally negative, we can reduce the penalty for semantically close ones. Two common strategies are:

- *Removal*: exclude the top-k most similar captions from the negative set, so they do not appear in the denominator.
- *Down-weighting* [12]: keep all captions in the denominator, but assign smaller weights for those that are semantically close or from the same class.

Both strategies soften the loss and the model will not be penalized as strongly for separating captions that might be, in our case, valid semantic alternatives, reducing the false negative problem when EEG responses cannot fully discriminate between similar stimuli.
Weighted contrastive. Let w_{ij} be a weight for how strongly negative j should be treated relative to anchor i (e.g., $w_{ij} = 0$ for removal, $w_{ij} < 1$ for down-weighting). The weighted InfoNCE loss is:

$$\mathcal{L}_{\text{debias}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(Z_{ii})}{\sum_{j=1}^{N} w_{ij} \exp(Z_{ij})}.$$

You should think about how to calculate w_{ij} to be a similarity-aware weight: captions with high similarity to get smaller w_{ij}, while dissimilar captions to keep $w_{ij} \approx 1$.

5.5.5 Category-Level Supervision (Cross-Entropy)

Attach a classifier to the EEG representation for $C = 20$ categories with logits $\mathbf{u}$ and label y:

$$\mathcal{L}_{\text{CE}} = -\log \frac{\exp(u_y)}{\sum_{c=1}^{C} \exp(u_c)}.$$

This can encourage category structure.

5.5.6 Combined Objective

You may start with knowledge distillation as a warm-up, and then add other contrastive loss to shape the geometry of the embedding space so that same-class items are close together. You can experiment with any combination:

$$\mathcal{L}_{\text{total}} = \lambda_{\cos}\mathcal{L}_{\cos} + \lambda_{\text{KD}}\mathcal{L}_{\text{KD}} + \lambda_{\text{debiased}}\mathcal{L}_{\text{debiased}} + \lambda_{\text{CE}}\mathcal{L}_{\text{CE}}.$$

5.5.7 Training Strategies for CLIP Integration

We will consider different strategies for aligning EEG signals with CLIP embeddings:

$$\text{Text Encoder Options:} \begin{cases} \text{Frozen} \\ \text{Partial Unfreezing} \\ \text{Adapters} \\ \text{LoRA} \end{cases}$$

- **Both models frozen:** You can start by experimenting caption retrieval when both EEG encoder and CLIP's text encoder are frozen, while training only the projection for EEG embedding into CLIP's space.
- **Frozen CLIP:** CLIP remains fixed. Only the EEG encoder and projection head are trained. This preserves CLIP's pretrained representations while adapting EEG signals to fit them.
- **Partial Unfreezing:** The earlier layers of the CLIP text encoder remain frozen, while the last one or two transformer layers and/or the text projection layer are unfrozen. This allows modest adaptation to EEG signals while retaining most of the pretrained structure.
- **Adapters:** Lightweight bottleneck modules are inserted inside CLIP's transformer layers. During training, only the adapters and the EEG encoder are updated, while the original CLIP parameters remain frozen.
- **LoRA (Low-Rank Adaptation):** LoRA was proposed by Hu et al. [13] to reduce the number of trainable parameters. Instead of adding separate modules, LoRA injects low-rank update matrices directly into the frozen weight matrices of CLIP's attention layers. For a frozen weight $W \in \mathbb{R}^{d\times d}$, LoRA learns a low-rank decomposition:

$$W' = W + BA, \quad B \in \mathbb{R}^{d\times r},\, A \in \mathbb{R}^{r\times d}, \quad r \ll d.$$

You will experiment with both **frozen** and partially **unfrozen** configurations. In addition, you will test at least one **parameter-efficient fine-tuning** method: *Adapters* or *LoRA*. For each configuration, track the number of trainable parameters to understand the trade-off between model capacity and computational efficiency. You will be provided with a jupyter notebook to explore some of these strategies. Feel free to use it as a starter code.

5.6 Evaluation Metrics

To assess how well EEG-derived embeddings align with CLIP's caption space, you will **repeat the analysis** from section 5.4.3 and report the values for EEG-Caption Retrieval.

6 Feasibility and Insights

This project is best viewed as an exploratory study. Because EEG data are inherently noisy and limited, caption retrieval results may be variable, which reflects the challenges of aligning brain signals with text embeddings. Meaningful progress in this area may depend on larger, more diverse EEG datasets and on novel preprocessing methods that explicitly address inter-subject variability and cross-session differences. Such advances could help transform feasibility studies into practical pipelines for brain-language alignment.

7 Conclusion

In this project, you first built and trained an EEG encoder for classification of EEG signals, establishing a baseline for how well neural activity can be decoded into discrete categories. You then explored CLIP's image–text embedding space, testing how well captions align with their corresponding images. You then investigated the feasibility of mapping EEG signals into the same space for caption retrieval. During the process, you worked with EEG encoders, projection strategies, and fine-tuning methods, and you practiced evaluating models with different metrics such as CLIPScore, Recall@K and MAP.

The results may vary according to the training choices. The aim is to engage with a challenging cross-modal problem, gain hands-on experience with advanced techniques (knowledge distillation, contrastive learning, parameter-efficient fine-tuning), and critically analyze the results. These experiences provide a solid foundation for future work in multimodal learning.

References

[1] Yi Wang, Jing Zhang, Chen Li, Hao Liu, Xiang Li, Xiaoxiao Wu, Han Zhao, Wei Chen, et al. A multi-subject and multi-session eeg dataset for modelling human visual object recognition. *Scientific Data*, 12(1):843, 2025.

[2] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti Hämäläinen. Mne software for processing meg and eeg data. *NeuroImage*, 86:446–460, 2014.

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, ICML '21, pages 8748–8763, 2021.

[4] Haoyu Song, Li Dong, Weinan Zhang, Ting Liu, and Furu Wei. Clip models are few-shot learners: Empirical studies on vqa and visual entailment. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6088–6100, Dublin, Ireland, 2022. Association for Computational Linguistics.

[5] Shubham Mishra et al. Image synthesis with graph conditioning: Clip-guided diffusion models for scene graphs. *arXiv preprint arXiv:2401.14111*, 2024.

[6] Anh Nguyen et al. Robotic-clip: Fine-tuning clip on action data for robotic applications. *arXiv preprint arXiv:2409.17727*, 2024.

[7] Weijie Zhao et al. Clip in medical imaging: A survey. *arXiv preprint arXiv:2312.07353*, 2023.

[8] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, ICLR '20, 2020.

[9] Jianping Gou, Baosheng Yu, Hongyu Li, et al. A comprehensive survey on knowledge distillation: Taxonomy, challenges, and future directions. *arXiv preprint arXiv:2503.12067*, 2025.

[10] Meng Liu, Jiacheng Wang, Yifan Lu, et al. Clip-kd: An empirical study of clip model distillation. *arXiv preprint arXiv:2307.12732*, 2023.

[11] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[12] Haochen Li, Xin Zhou, Anh Tuan Luu, and Chunyan Miao. Rethinking negative pairs in code search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 12760–12774, 2023.

[13] Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.