Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Artificial Intelligence

Project Lab Report 1

EEG-to-Text Translation Using the Thought2Text Framework and Large Language Models

Written by:	Balkis Karoui
Neptun:	BDQD5E
Field:	Computer Engineering
Specialization:	Artificial Intelligence
E-mail:	Balkis.Karoui@edu.bme.hu

Supervisor:	Mohammed Salah Al-Radhi
E-mail:	malradhi@tmit.bme.hu

Academic year: 2025/2026 (Fall semester)

Abstract

This project explores how brain signals (EEG) can be translated into natural text using state-of-the-art large language models. The baseline is the Thought2Text framework, which combines EEG encoders, multimodal alignment, and LLM fine-tuning. In this project, we will experiment with the DeepSeek-1.5B model instead of the original Mistral-7B to see how model choice impacts accuracy and descriptiveness. The goal of this project is to extend and evaluate the framework using DeepSeek-1.5B, a recently released instruction-tuned LLM, and to analyze its performance against the published Mistral-7B-Instruct baseline.

The project will begin by replicating the Thought2Text baseline pipeline. Next, the DeepSeek-1.5B model will be integrated into the architecture. Advanced experiments will focus on prompt optimization and cross-subject evaluation. Performance will be evaluated through standard natural language generation metrics such as BLEU, ROUGE ,SBERT-Similarity, and BERTScore, along with qualitative analysis of generated outputs.

This research lies at the intersection of brain–computer interfaces (BCI) and large language models, aiming to provide insights into how modern instruction-tuned LLMs can improve brain-signal decoding.

Keywords: EEG-to-Text, Large Language Models, DeepSeek-1.5B, Thought2Text, Text Generation, Instruction Tuning, Natural Language Processing, Neural Decoding.

1 Introduction

1.1 Theoritical background

Brain–Computer Interface (BCI) systems aim to build a direct connection between brain activity and external devices. They allow a person to send commands or information without using muscles or speech. This idea has been studied for many years in neuroscience, engineering, and human–computer interaction. Today, BCIs are used in several areas such as assistive communication for patients with paralysis, neurorehabilitation, gaming, and cognitive monitoring. Among different types of brain signals, electroencephalography (EEG) is one of the most common methods because it is non-invasive, affordable, and portable. EEG provides high temporal resolution, meaning it can capture rapid changes in brain activity, but it has low spatial resolution and a high level of noise, which makes interpretation difficult [1].

Despite its limitations, EEG remains attractive for BCI research. The difficulty comes from the nature of the signal itself: electrical activity recorded on the scalp reflects the sum of many neural sources, mixed together and affected by the skull, scalp, and environmental noise. This results in signals that are weak, noisy, and highly variable. The variability also appears between subjects, since each person's brain structure and cognitive responses are different. Even the same person may show different EEG patterns across sessions due to attention, fatigue, or electrode placement issues. For this reason, developing robust EEG decoding systems has always been a major challenge.

Over the last decade, deep learning methods have significantly changed how EEG data can be processed. Earlier approaches relied on handcrafted features such as frequency bands, wavelets, or spatial filters. These traditional methods had limited flexibility and struggled to generalize to new data. With the rise of convolutional neural networks (CNNs), recurrent models, and attention-based models, researchers began to learn features directly from raw or minimally processed EEG signals. This shift made it possible to perform tasks such as emotion recognition, cognitive workload prediction, motor imagery classification, and visual recognition with improved accuracy. Multimodal learning, where EEG is combined with other data types like images or text, further expanded what could be achieved.

In parallel, natural language processing (NLP) has seen major advances due to the emergence of large language models (LLMs). Models such as GPT-4, LLaMA, Mistral, and Qwen are trained on enormous text corpora and can generate coherent and meaningful text across many domains. These models also support instruction tuning, meaning they can follow prompts like "describe this image" or "summarize the text." More recently, multimodal extensions of LLMs have allowed models to work with images, audio, and video. This opens an interesting possibility for BCI research: instead of designing a separate decoder for EEG-to-text generation, we can try to map EEG signals into the embedding space of an LLM and let the LLM produce the actual language output.

The Thought2Text framework by Mishra et al. (2025) is one of the first complete systems that attempts to generate natural-language descriptions directly from EEG signals using an instruction-tuned LLM. The authors build their method around a public EEG dataset collected from six subjects. Each subject looked at a set of 50 images from 40 different object categories (pages 1–3 of the paper). While the subjects viewed these images, EEG signals were recorded using 128 channels for about 0.5 seconds per image. To create training labels, captions for each image were generated by GPT-4 and checked by human annotators for fluency and correctness. This produced a tri-modal dataset with aligned `<EEG, Image, Text>` samples, which is essential for training a multimodal model.

A key idea in the Thought2Text paper is the use of visual stimuli instead of written text. When people read, the brain goes through many complex steps such as recognizing letters, converting them into sounds, understanding grammar, and processing meaning. These processes happen at different times and involve several overlapping brain regions. This makes it difficult to align EEG signals with the exact words being processed. In contrast, visual perception is more direct and instinctive. When a person sees an object, the brain quickly extracts features such as shape, texture, or color. Because of this, EEG from visual stimuli can provide a more stable and language-independent signal for decoding.

The Thought2Text method is built on three main stages. In Stage 1, the system trains an EEG encoder based on ChannelNet to extract meaningful embeddings from the multichannel EEG input. The encoder is trained with a dual objective: (1) predict the object category shown in the image using a classifier, and (2) align the EEG embedding with the CLIP image embedding using mean squared error (MSE). CLIP is a vision-language model that learns image-text alignment and produces high-quality image embeddings. By aligning EEG embeddings with CLIP embeddings, the system forces the EEG encoder to learn features that represent the main semantic content of the visual stimulus (Figure 1, page 4).

Stage 2 focuses on preparing the LLM to work with multimodal embeddings. LLMs normally accept only text as input, so they cannot directly process EEG or image embeddings. To solve this, the authors introduce a projector, which is a small neural network that maps CLIP image embeddings into the token embedding space of the LLM. These projected embeddings are inserted into the model in place of a special token such as `<image>`. During this stage, the LLM stays frozen, and only the projector is trained. This helps the model learn how visual information should influence the generated text and creates a smooth pathway for multimodal alignment.

Stage 3 is similar to Stage 2, but instead of CLIP embeddings, the projector is trained using EEG embeddings from Stage 1. The EEG encoder and LLM remain frozen during this process. This design decision avoids instability because EEG data is noisy and difficult to fine-tune on. The projector learns how EEG embeddings relate to the linguistic space of the LLM. After Stage 3 is complete, the LLM can generate text based solely on EEG input.

During inference, the system uses only the EEG signal and a basic language prompt that includes the predicted object label. The object label is obtained from the classifier in Stage 1. The LLM then generates a short natural-language description that matches the image the subject saw (pages 5–6 of the paper). To evaluate the generated text, standard natural language generation metrics such as BLEU, ROUGE, METEOR, and BERTScore are used. The authors also perform qualitative evaluation using GPT-4 to judge fluency and adequacy.

The Thought2Text framework is notable because it combines EEG decoding, visual-semantic alignment, and large language models in a single pipeline. Earlier EEG-to-text systems often used traditional transformers or RNNs and required large datasets or carefully designed feature extraction methods. In contrast, Thought2Text benefits from the strong generalization ability of LLMs and the semantic richness of CLIP embeddings. The method shows that even small EEG datasets can be used to train a model capable of producing meaningful textual descriptions.

Overall, the theoretical background for this project includes several key areas: principles of EEG and BCI technology, challenges in neural decoding, multimodal representation learning, image-text alignment, and LLM conditioning through learned embedding spaces. The Thought2Text model provides a strong foundation for exploring alternative LLMs, such as DeepSeek-1.5B, which may offer better efficiency while still producing high-quality text. Understanding these concepts is essential for replicating, modifying, and extending the system as part of this project.

1.2 State of the Job at the Beginning of the Semester

At the beginning of the semester, several materials, resources, and initial preparations were already available for starting the project. These included theoretical knowledge, learning resources, and the main research paper that defines the framework used in this work. The following items summarize the state of the job before the new semester activities began:

- The Thought2Text research paper [1], including its full methodology and model design.
- Initial understanding of EEG-based decoding tasks and their challenges.
- Access to the public CVPR2017 EEG dataset used in the Thought2Text framework.
- Basic familiarity with deep learning and neural networks from previous courses.

- Preliminary study of Convolutional Neural Networks (CNNs), based on recommendations to understand the EEG encoder used in the paper.
- Enrollment in or auditing of the Coursera "Convolutional Neural Networks" course for strengthening theoretical background.
- Watching the 3Blue1Brown "Neural Networks" video series to gain an intuitive understanding of network operations.
- Initial review and breakdown of the multi-stage training pipeline used in Thought2Text.
- Availability of the development environment (Python, PyTorch, Jupyter Notebook) required for reproducing the paper's baseline workflow.

These resources formed the foundation for the work completed during the semester and helped separate prior preparation from the new contributions developed in this project.

2 Methodology

2.1 System Design and Architecture

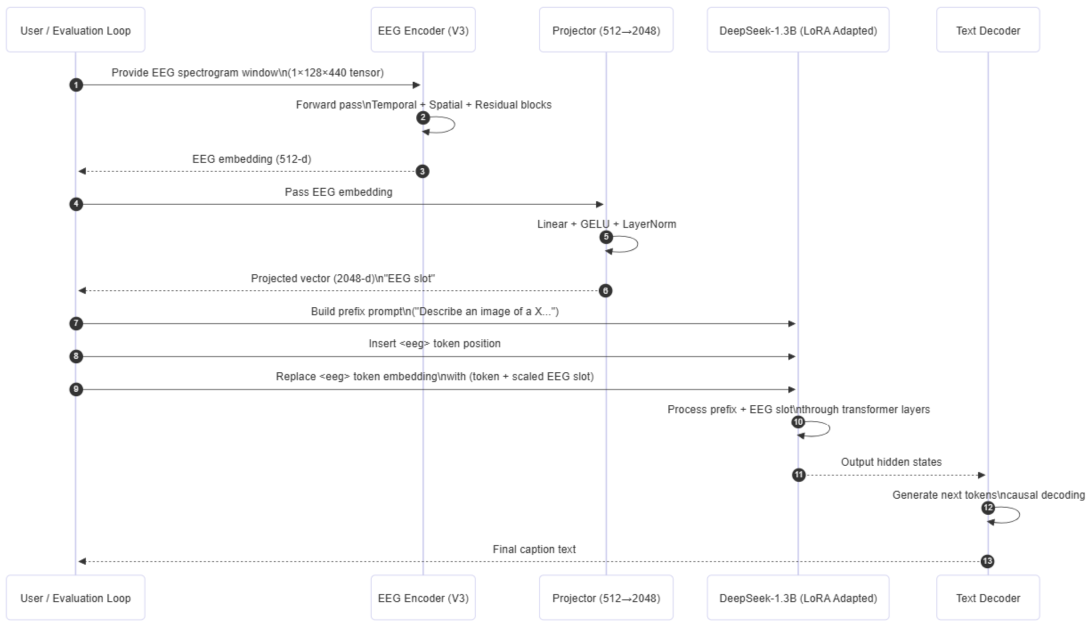

User / Evaluation Loop
EEG Encoder (V3)
Projector (512→2048)
DeepSeek-1.3B (LoRA Adapted)
Text Decoder
Provide EEG spectrogram window\n(1×128×440 tensor)
Forward pass\nTemporal + Spatial + Residual blocks
EEG embedding (512-d)
Pass EEG embedding
Linear + GELU + LayerNorm
Projected vector (2048-d)\n"EEG slot"
Build prefix prompt\n("Describe an image of a X...")
Insert <eeg> token position
Replace <eeg> token embedding\nwith (token + scaled EEG slot)
Process prefix + EEG slot\nthrough transformer layers
Output hidden states
Generate next tokens\ncausal decoding
Final caption text

Figure 1: sequence diagram of the encoder–projector–LLM pipeline.

The sequence diagram in Figure 1 summarizes the inference pipeline used in the final EEG-to-text system. The goal of the architecture is to transform an EEG spectrogram into a short natural-language caption by combining the EEG encoder, the multimodal projector, and the DeepSeek language model.

2.2 Data Processing and Preparation

Before training the models, the dataset had to be explored and organized. The data consisted of three elements: EEG spectrogram windows, sketch images of the stimuli, and text captions describing the images. All files were linked by a shared trial identifier called `base_id`. One trial represented one image that a subject viewed while EEG was recorded.

2.2.1 Dataset structure.

Each ImageNet class folder contained several trials. For every trial, there were:

- 5–6 EEG spectrogram windows,
- one sketch image,
- one caption file.

2.2.2 Index creation

A dataframe was built to index all EEG windows and match them with the correct sketch image, caption file, class label, and trial ID. Duplicate entries were removed, and basic checks confirmed that all trials had both sketches and captions.

2.2.3 Caption loading and class names

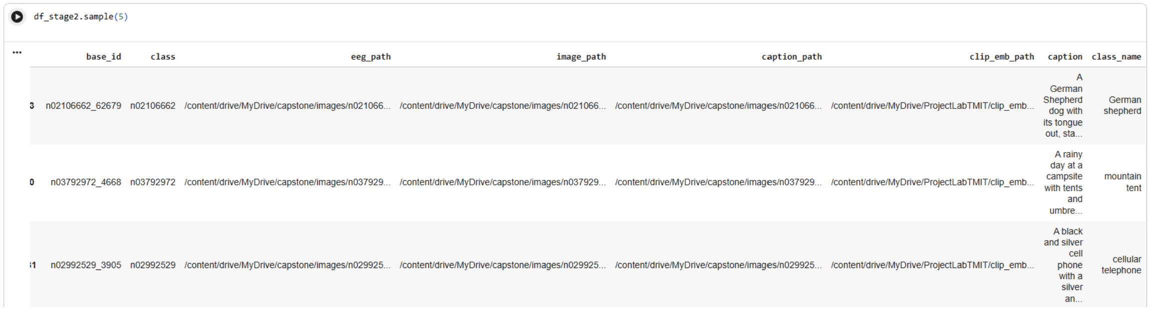
The caption text was loaded from the files and stored directly in the dataframe to simplify training. The ImageNet class IDs were also converted into readable class names using WordNet.

2.2.4 Final dataset.

After processing, the dataset contained:

- 40 classes,
- 1996 unique trials,
- 23,930 EEG spectrogram windows,
- complete alignment between EEG, sketch, and caption.

The final dataframe was saved and used as the main input for Stage 1, Stage 2, and Stage 3 of the training pipeline. A small sample of the dataframe is shown in Figure X to illustrate its structure.



```
df_stage2.sample(5)
```

	base_id	class	eeg_path	image_path	caption_path	clip_emb_path	caption	class_name
3	n02106662_62679	n02106662	/content/drive/MyDrive/capstone/images/n021066...	/content/drive/MyDrive/capstone/images/n021066...	/content/drive/MyDrive/capstone/images/n021066...	/content/drive/MyDrive/ProjectLabTMIT/clip_emb...	A German Shepherd dog with its tongue out, sta...	German shepherd
0	n03792972_4668	n03792972	/content/drive/MyDrive/capstone/images/n037929...	/content/drive/MyDrive/capstone/images/n037929...	/content/drive/MyDrive/capstone/images/n037929...	/content/drive/MyDrive/ProjectLabTMIT/clip_emb...	A rainy day at a campsite with tents and umbre...	mountain tent
1	n02992529_3905	n02992529	/content/drive/MyDrive/capstone/images/n029925...	/content/drive/MyDrive/capstone/images/n029925...	/content/drive/MyDrive/capstone/images/n029925...	/content/drive/MyDrive/ProjectLabTMIT/clip_emb...	A black and silver cell phone with a silver an...	cellular telephone

Figure 2: Preview of the final multimodal dataframe containing EEG, sketch image paths, caption text, and class information.

2.3 Stage 1: EEG Encoder Training

The goal of Stage 1 was to train an EEG encoder that can produce meaningful embeddings from EEG spectrogram windows. These embeddings should be aligned with CLIP image embeddings, since CLIP provides a strong semantic representation of the visual stimulus. At the same time, the encoder must also predict the image class, which gives an additional supervision signal and helps the model learn more discriminative features.

2.3.1 Training objective.

The original Thought2Text paper trained the encoder using a combination of two losses:

$$L_{\text{original}} = (1-\alpha)\,\text{MSE}(h_{\text{EEG}}, h_{\text{CLIP}}) + \alpha\,\text{CE}(y_{\text{pred}}, y_{\text{true}}).$$

The **Mean Squared Error (MSE)** term measures the squared distance between the EEG embedding and the CLIP embedding. A lower MSE means that the EEG embedding is numerically closer to the CLIP embedding.

The **Cross-Entropy (CE)** loss is used for classification. It measures how well the predicted class distribution matches the true class label. A lower CE value means the encoder is correctly identifying the ImageNet class more often.

In our use-case, the MSE-based alignment did not give strong results. The EEG embeddings did not converge well toward the CLIP embeddings, and cosine similarity during validation stayed low. Because of this, we modified the loss function and replaced MSE with cosine similarity:

$$L = L_{\text{cosine}}(h_{\text{EEG}}, h_{\text{CLIP}}) + \lambda\,L_{\text{CE}},$$

where a small coefficient ($\lambda = 0.05$) controls the weight of the classification term.

Cosine loss focuses on the *direction* of the embedding vectors instead of their magnitude. This worked significantly better because both EEG and CLIP embeddings were normalized and cosine distance is a more stable measure for alignment in embedding spaces.

2.3.2 Qualitative Results

After training the EEG encoder for ten epochs, two types of evaluation were performed: classification accuracy and cosine similarity between the predicted EEG embeddings and the CLIP image embeddings.

Classification accuracy. The encoder reached a validation accuracy of approximately **18%**. This is lower than the values reported in the original Thought2Text paper (around 40–56%), but this difference is expected. The paper trained on the full dataset and used a carefully tuned ChannelNet implementation, while our version was a simplified reproduction with reduced training time and fewer optimization steps. Even with lower accuracy, the encoder still learned useful patterns and general class distinctions.

Cosine similarity. The alignment between EEG embeddings and CLIP embeddings was evaluated using cosine similarity. The average validation cosine similarity obtained was:

$$\text{Average EEG} \rightarrow \text{CLIP cosine similarity} = \mathbf{0.7726}.$$

This value shows that the encoder was able to produce embeddings that matched the direction of the CLIP embeddings reasonably well. The performance also confirms that switching from MSE loss to cosine loss helped stabilize training and improved alignment.

2.4 Stage 2: Projector Training

Stage 2 follows the structure of the original Thought2Text pipeline: the goal is to learn a mapping from CLIP image embeddings to the token embedding space of a large language model (LLM). During this stage, the model does not use EEG signals. Instead, it learns how a visual embedding should modify the LLM's internal representation so that it can produce a correct caption.At this stage, only the projector is trained. The llm stays completely frozen.

2.4.1 Using a smaller language model.

The original paper uses Mistral-7B as the LLM backbone. For computational reasons, this project uses a much smaller open-source model, **DeepSeek-Coder-1.3B-Instruct**, which fits comfortably on a single GPU under 8-bit quantization. This change required several adaptations:

- DeepSeek uses a **different chat template** and special tokens such as `<|User|>` and `<|Assistant|>`. Therefore, a custom text-generation prefix had to be built manually to match DeepSeek's expected input format.
- The token embedding dimension of DeepSeek (2048) differs from that of Mistral-7B, so the projector architecture was adjusted accordingly.
- Some prompt-building tools from HuggingFace do not automatically handle DeepSeek's conversational style, so the prefix construction and tokenization were implemented from scratch.

2.4.2 Projector architecture.

The projector (`Projector_V1`) is a small feed-forward network:

- input: 512-dimensional CLIP image embedding,
- two linear layers with GELU activations and LayerNorm,
- output: a vector in the LLM embedding space (2048 dimensions).

2.4.3 Training objective.

Training is performed by inserting the projected image embedding into the middle of the input sequence:

$$\text{(prefix tokens)} + \text{(projected CLIP embedding)} + \text{(caption tokens)}.$$

All prefix tokens and the multimodal embedding are masked with label -100, meaning that the LLM is only trained to predict the caption tokens.

The loss is simply the standard language-modeling loss:

$$L_{\text{stage2}} = \text{CE}(\hat{y}, y),$$

where $\hat{y}$ is the LLM output and y is the ground-truth caption.

2.4.4 Qualitative Results.

After two epochs, the projector was able to produce meaningful conditioning signals. During inference, the LLM generated visually relevant captions based only on the projected CLIP embedding. Even with the small 1.3B model, the text quality was coherent, and the conditioning effect was visible. This shows that the Stage 2 training pipeline works even with a much smaller LLM than the original paper.

2.5 Stage 3: EEG-to-Text Training

Stage 3 is the final step of the Thought2Text pipeline, where the EEG encoder and the multimodal projector are connected to the language model so that the model can generate a caption directly from an EEG signal. In this stage, the model no longer uses CLIP embeddings during inference. Instead, the EEG encoder output is projected into the LLM embedding space and inserted into the token sequence as a learned multimodal feature.

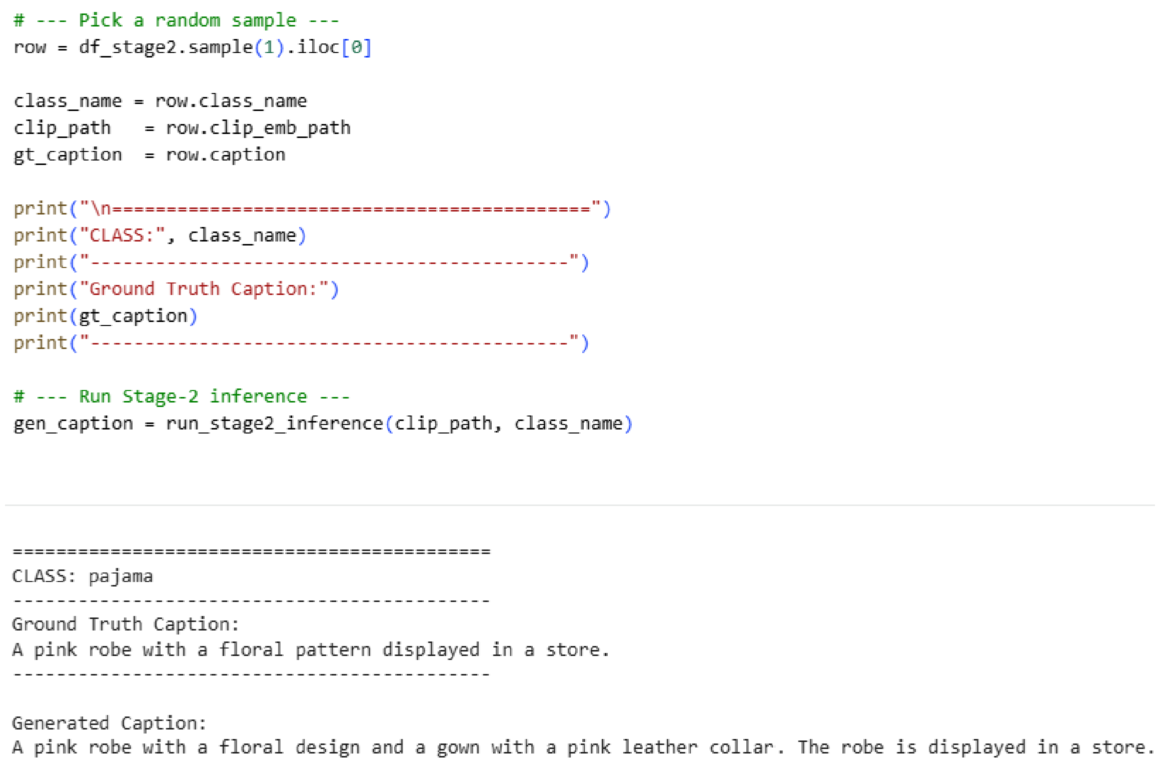
2.5.1 Training objective.

To train this joint system, we used a combined loss function:

$$L = \lambda_{\text{caption}} \cdot CE(\hat{y}, y) + \lambda_{\text{clip}} \cdot (1 - \cos(h_{\text{EEG}}, h_{\text{CLIP}}))\,.$$

The first term (**caption loss**) trains the LLM to reproduce the ground-truth caption. The second term (**alignment loss**) ensures that the EEG embedding remains close to the CLIP image embedding.

Both losses are important: the first teaches language generation, the second stabilizes the EEG encoder.



```
# --- Pick a random sample ---
row = df_stage2.sample(1).iloc[0]

class_name = row.class_name
clip_path   = row.clip_emb_path
gt_caption  = row.caption

print("\n============================================")
print("CLASS:", class_name)
print("--------------------------------------------")
print("Ground Truth Caption:")
print(gt_caption)
print("--------------------------------------------")

# --- Run Stage-2 inference ---
gen_caption = run_stage2_inference(clip_path, class_name)
```

```
============================================
CLASS: pajama
--------------------------------------------
Ground Truth Caption:
A pink robe with a floral pattern displayed in a store.
--------------------------------------------

Generated Caption:
A pink robe with a floral design and a gown with a pink leather collar. The robe is displayed in a store.
```

Figure 3: Example output from Stage 2 showing the ground-truth caption and the LLM-generated caption using the projected CLIP embedding.

2.5.2 Dataset setup.

The Stage 3 dataset reuses the same samples as Stage 2, but now includes:

- the EEG spectrogram,
- the CLIP image embedding (for alignment loss),
- a DeepSeek chat-style prefix prompt,
- the ground-truth caption.

This allows the model to learn both semantic alignment and caption generation.

2.5.3 Injecting EEG information into the LLM.

The original Thought2Text paper used a special multimodal token inside Mistral-7B. DeepSeek-1.3B, however, does not include such a mechanism. This required implementing a custom method to **manually** insert EEG features into the LLM's hidden embedding sequence.

Three challenges appeared:

1. locating the correct embedding layer inside the DeepSeek architecture,

2. matching the EEG projector output dimension to DeepSeek's hidden size (2048),
3. scaling the EEG embedding so that it fits the statistical distribution of token embeddings.

To solve this, a new special token `<eeg>` was added to the tokenizer, and its embedding was replaced by:

$$e_{\text{EEG}} = e_{\text{token}} + s \cdot P(E_{\text{EEG}}),$$

where P is the projector and s is a scaling factor that matches the variance of the LLM embedding matrix. This "EEG slot" acts as a multimodal anchor inside the prompt.

2.5.4 Iterative development (Tests 1–3).

Finding the correct injection point required several experimental rounds:

Test 1: Direct replacing of a token embedding. The EEG vector was added directly to the embedding of a special token. However, DeepSeek uses a nested architecture, and the first attempt targeted the wrong layer. This produced unstable results and inconsistent sequence lengths.

Test 2: Correct embedding layer but wrong tensor shape. The next attempt found the right embedding layer, but the EEG slot had mismatched dimensions (sometimes $(1, 2048)$ instead of $(1, 1, 2048)$). This caused silent broadcast errors and incorrect attention masking.

Test 3: Final working solution. A final version fixed:

- embedding-layer location,
- dimensionality $(1 \times 1 \times 2048)$,
- statistical scaling (matching token embedding variance),
- prefix–EEG–caption concatenation.

This version produced stable training and coherent EEG-conditioned captions.

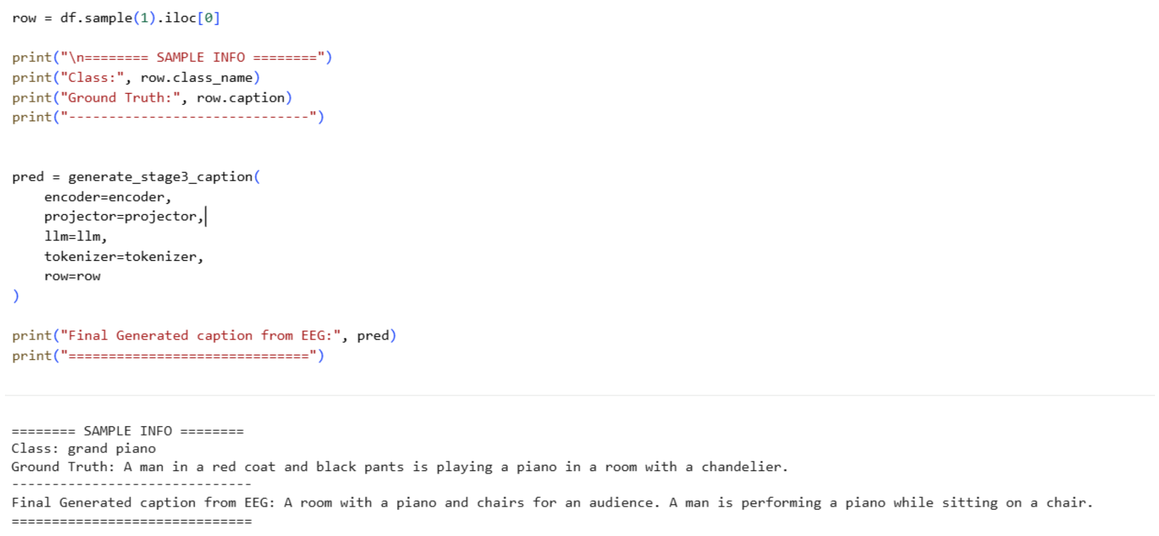
2.5.5 LLM fine-tuning with LoRA.

Because DeepSeek-1.3B is small, it benefits from light adaptation. A low-rank adaptation (LoRA) was applied only to the attention projection layers. This allowed the LLM to learn how to use the EEG slot without modifying the entire language model.

Only a subset of the dataset (1500 samples) was used to keep computational cost reasonable.

2.5.6 Qualitative Results.

After three epochs, the system was able to generate short factual captions conditioned only on EEG. The captions followed the expected structure and often included the correct class name. Although the model is much smaller than the original Mistral-7B used in the paper, the qualitative results show that the EEG embeddings were successfully injected into the LLM's hidden representation and influenced the generated text.



```
row = df.sample(1).iloc[0]

print("\n======== SAMPLE INFO ========")
print("Class:", row.class_name)
print("Ground Truth:", row.caption)
print("------------------------------")


pred = generate_stage3_caption(
    encoder=encoder,
    projector=projector,
    llm=llm,
    tokenizer=tokenizer,
    row=row
)

print("Final Generated caption from EEG:", pred)
print("==============================")
```

```
======== SAMPLE INFO ========
Class: grand piano
Ground Truth: A man in a red coat and black pants is playing a piano in a room with a chandelier.
------------------------------
Final Generated caption from EEG: A room with a piano and chairs for an audience. A man is performing a piano while sitting on a chair.
==============================
```

Figure 4: Example output from Stage 3 showing the ground-truth caption and the caption generated directly from EEG using the trained encoder–projector–LLM pipeline.

3 Quantitative Results

3.1 Metric Interpretation

Statistic	BLEU	ROUGE-L	BERTScore	SBERT-Similarity
count	50.000	50.000	50.000	50.000
mean	0.2805	0.3843	0.9118	0.6403
std	0.2199	0.1526	0.0291	0.2043
min	0.0097	0.1395	0.8597	0.2618
25%	0.1110	0.2614	0.8915	0.4758
50%	0.2691	0.3858	0.9086	0.6790
75%	0.3806	0.4604	0.9369	0.7910
max	0.9216	0.7879	0.9663	0.9401

Table 1: Global metric summary for 50 generated captions

The table above presents the global results for 50 randomly evaluated classes. The average BERTScore (0.91) and SBERT similarity (0.64) indicate that most generated captions are semantically close to the ground-truth captions. This means that the model usually identifies the correct concept or object category from the EEG input. In contrast, BLEU (0.28) and ROUGE-L (0.38) remain modest. These metrics measure exact word or phrase overlap, and therefore suggest that the model often produces paraphrased sentences rather than matching the reference text word-by-word.

3.1.1 Distribution Across the Dataset

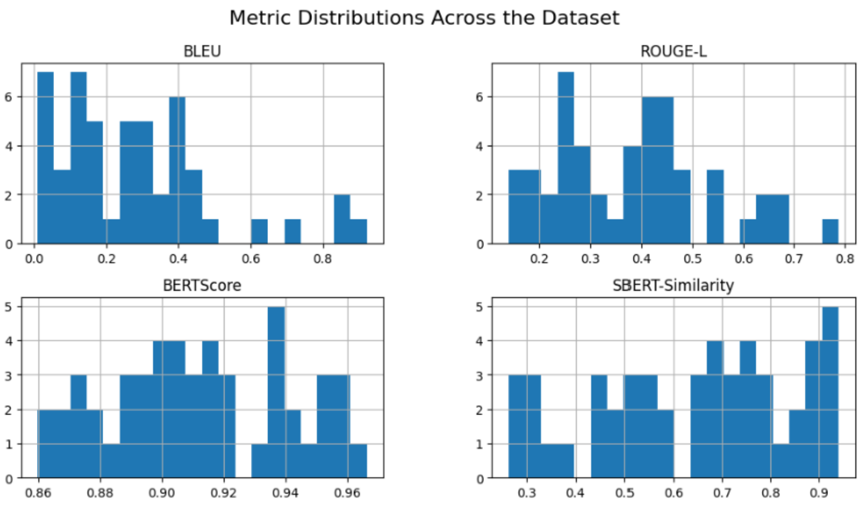


Figure 5: Distribution of evaluation metrics (BLEU, ROUGE-L, BERTScore, and SBERT-Similarity) across the dataset.

The metric distributions support this interpretation. BLEU and ROUGE-L show wide variability, with many samples scoring low but a few reaching high values. This indicates that the model sometimes matches the reference phrasing closely, but not consistently. On the other hand, BERTScore has a narrow distribution around 0.90–0.93, and SBERT similarity also shows a strong cluster above 0.70. These patterns confirm that semantic alignment is much more stable across the dataset than lexical overlap.

3.1.2 Performance by Class

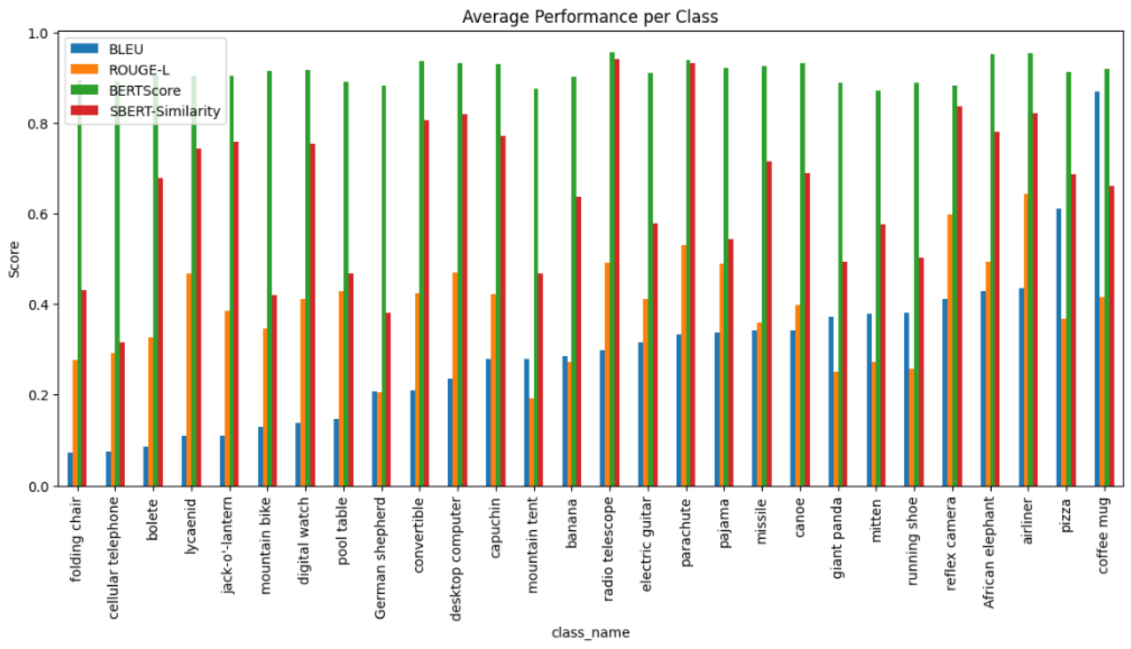


Figure 6: Performance of the EEG-to-text model across different object classes

When analysing performance by class, we observe that classes with distinctive visual identities (e.g., highly recognizable objects) tend to achieve higher semantic scores. More ambiguous classes, or classes where the EEG embeddings are less separable, show lower similarity. This behaviour is expected because the EEG signals are noisy and the dataset is small. As a result, some classes are easier for the encoder and projector to map into a meaningful embedding space, while others remain challenging.

4 conclusion

In summary, the experiment shows that the system is able to generate captions that are semantically accurate, even if the wording is not always the same as the reference caption. The lower BLEU and ROUGE scores are mainly due to the language-model component: our setup uses a smaller LLM and a simple way of inserting EEG information, so producing the exact same words as the ground truth is naturally more difficult.

However, considering the limited compute resources and the much smaller model size, the system still achieves results that are reasonably close to those reported in the original Mistral-7B paper. This suggests that the approach is promising, and future work could focus on using larger language models or more advanced fusion techniques to improve lexical accuracy while keeping the strong semantic alignment.

References

[1] A. Mishra, S. M. Shurid, et al., *Thought2Text: Text Generation from EEG Signal using Large Language Models (LLMs)*, arXiv preprint arXiv:2410.07507, 2024.

[2] DeepSeek-AI, *DeepSeek-LLM, DeepSeek-Coder and DeepSeek-Instruct Models*, GitHub Repository, 2024. Available: `https://github.com/deepseek-ai`

[3] DeepSeek-AI, *DeepSeek Model Documentation and Chat Template Specification*, 2024. Available: `https://huggingface.co/deepseek-ai`

[4] M. Verwoert, M. C. Ottenhoff, S. Goulis, et al., *Dataset of Speech Production in Intracranial Electroencephalography*, Scientific Data, vol. 9, no. 1, pp. 1–9, 2022.

[5] A. Ng, *Convolutional Neural Networks*, DeepLearning.AI, Coursera Online Course, 2024. Available: `https://www.coursera.org/learn/convolutional-neural-networks`

[6] T. Wolf, L. Debut, V. Sanh, et al., *Transformers: State-of-the-Art Natural Language Processing*

[7] DeepSeek-AI, *DeepSeek Documentation and Model Cards*, Available: `https://huggingface.co/deepseek-ai`

[8] DeepSeek-AI, *DeepSeek Official GitHub Repository*, 2024. Available: `https://github.com/deepseek-ai`

[9] HuggingFace, *Transformers Documentation*, Available: `https://huggingface.co/docs/transformers`

[10] HuggingFace, *Tokenizers: Fast Tokenization Library*, Available: `https://huggingface.co/docs/tokenizers`

[11] HuggingFace, *PEFT: Parameter-Efficient Fine-Tuning Library*, Available: `https://huggingface.co/docs/peft`

[12] Google, *Colab GPU/TPU Runtime Documentation*, Available: `https://research.google.com/colaboratory/faq.html#gpu-access`

[13] PyTorch Foundation, *PyTorch Documentation*, Available: `https://pytorch.org/docs/stable`