EEG-to-Text Translation Using the Thought2Text Framework

Wang Hongen (FT7J14)

Chen Jiayu (PUOOOZ)

Budapest University of Technology and Economics

Declaration of using LLMs

In the work titled *"EEG-to-Text Translation Using the Thought2Text Framework"*, we used Large Language Models (LLMs) to assist with the style correction of this document. We also used LLMs to help generate explanatory comments for the code in *inference.py* in order to better understand its functionality.

1. Introduction

Trying to decode human thoughts into words is one of the biggest challenges in AI, neuroscience, and brain-computer interface (BCI) research. Usually, BCIs have been used for simpler tasks—like detecting when someone wants to move a cursor or recognizing basic brain patterns. But now, with advances in multimodal AI and large language models (LLMs), we're getting closer to translating abstract thoughts—like the meaning behind what someone sees—into actual sentences. If we could pull this off, it would open doors for assistive tech, VR applications, and smarter neuroadaptive systems.

EEG stands out among noninvasive brain recording methods because it's cheap, portable, and easy to use in everyday settings. But it's also tricky: EEG signals are noisy, spatially blurry, and vary a lot from person to person. Past attempts to turn EEG into text often used small word sets or basic classifiers, which limited how much they could express. Some newer methods like EEG2TEXT aim for open vocabulary, but they don't really tap into the reasoning and context skills of today's LLMs. Because of that, generating good language straight from EEG is still largely unexplored.

One big issue is what you might call "language interference"—when someone reads or listens, their brain activity gets mixed up with language processing, making it hard to extract pure thought content. To avoid this, the Thought2Text approach records EEG while people look at pictures instead of text. By linking EEG patterns to CLIP-based image embeddings, and then using instruction-tuned LLMs, the system can learn to describe what someone sees directly from their brain signals. They used the CVPR2017 EEG dataset, which has image-EEG pairs recorded together, to train the model.

Thought2Text uses a three-step training pipeline: first, an EEG encoder is aligned to CLIP's visual space; second, the LLM is primed with image embeddings; third, a projection layer is fine-tuned so it can take EEG inputs during inference. This staged method helps—training end-to-end from EEG to text is unstable with limited noisy data, so using images as a middle step makes learning more reliable. The original results show it's possible to get coherent language from noninvasive EEG with modern multimodal models.

In our version, we made two main upgrades. First, we swapped Mistral-7B for DeepSeek-7B, which handles instructions better and produces richer text. Second, we added prompt-enhancement tricks to encourage more detailed and descriptive outputs from the LLM. Prompt engineering really does help—in our tests, the captions became more accurate and vivid even with the same EEG inputs.

By blending these upgrades into the Thought2Text pipeline, we're aiming for higher-quality EEG-to-text generation. Our approach keeps the core idea of aligning brain data with multimodal AI, but leans more on modern LLM capabilities. It's a step toward low-cost "thought-to-text" systems that might someday work in real-world BCIs.

2. Methods of Training

2.1 Description of the Chosen Neural Network

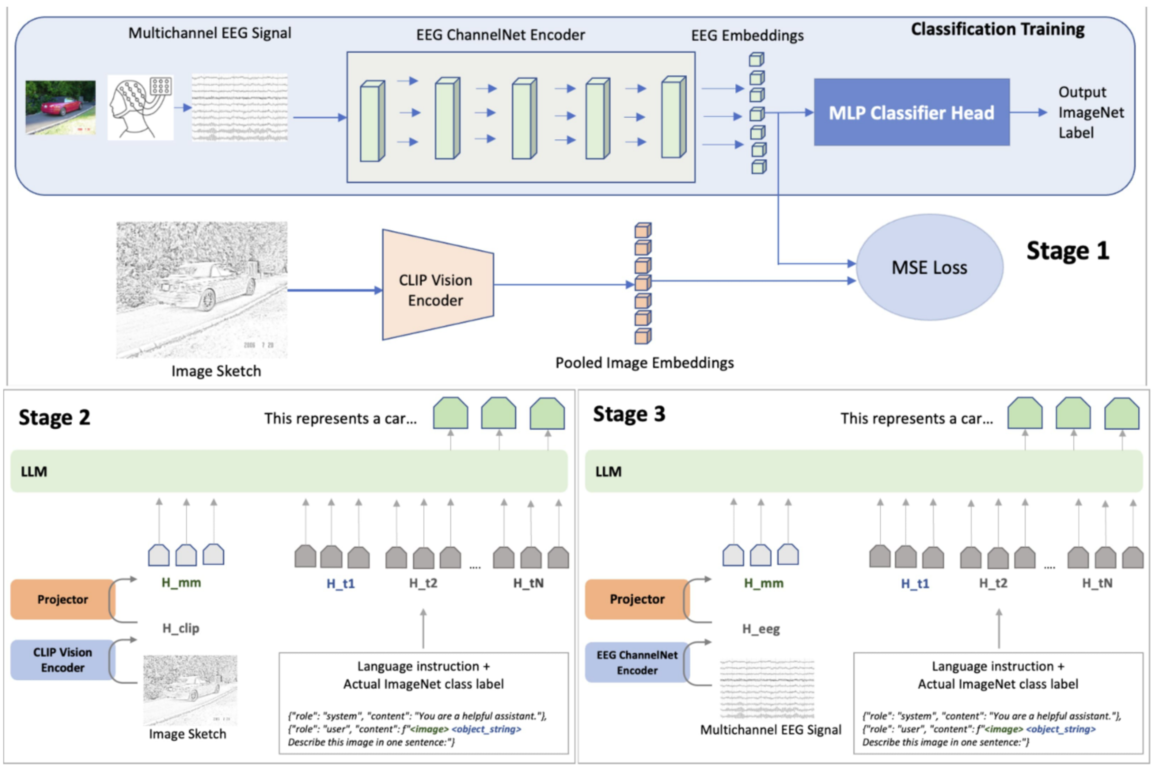
Our method builds on the multimodal EEG-to-text generation pipeline proposed in the Thought2Text framework, but extends it by using DeepSeek-7B as the language model and incorporating enhanced prompting strategies. The system consists of three major components: an EEG encoder, a multimodal projection layer, and a large language model. The EEG encoder is implemented using a ChannelNet-style multichannel convolutional neural network that processes 128-channel EEG signals cropped between 20–460 ms. It produces a 512-dimensional EEG embedding along with object classification logits. Training the encoder involves two supervision signals: an MSE loss that enforces alignment between EEG embeddings and CLIP image embeddings, and a cross-entropy loss that predicts the ImageNet-derived object label corresponding to the visual stimulus. This dual-loss design grounds the EEG embedding space in visual semantics and follows the alignment principles validated in the Thought2Text paper.

The language model backbone is DeepSeek-7B, chosen for its strong reasoning capabilities and ability to generate detailed, coherent descriptions. DeepSeek-7B replaces the Mistral-7B model used in the original framework. A multimodal projection layer maps EEG embeddings into the token embedding space of the LLM so they can be used as conditioning signals during generation. During training, the projection layer is the only component updated, while both the EEG encoder and the LLM remain frozen.

2.2 Architecture and Design Decisions

Our system follows the three-stage architecture defined in Thought2Text as shown in Figure 1. In Stage 1, the EEG encoder is trained using EEG–image pairs to learn an embedding space aligned with CLIP features while also supporting object classification. In Stage 2, the LLM is trained using CLIP image embeddings instead of EEG, enabling it to incorporate multimodal embeddings while preserving the stability of the frozen backbone. In Stage 3, EEG embeddings replace image embeddings, and only the multimodal projection layer is trained so the LLM can interpret EEG-driven inputs.

This staged design is necessary because EEG data is noisy and extremely limited compared to typical language model training datasets. Aligning EEG signals with CLIP embeddings provides a semantically meaningful intermediate representation, making downstream LLM training feasible. Freezing all components except the projection layer also prevents overfitting and instability. Additionally, we introduce enhanced instruction prompts during inference—such as detailed or creative prompts—to encourage the model to generate richer and more descriptive output.



```
{"role": "system", "content": "You are a helpful assistant."},
{"role": "user", "content": f"<image> <object_string>
Describe this image in one sentence:"}
```

```
{"role": "system", "content": "You are a helpful assistant."},
{"role": "user", "content": f"<image> <object_string>
Describe this image in one sentence:"}
```

Figure 1: Multi stage training process for Thought2Text solution.

2.3 Manual and/or Automatic Hyperparameter Optimization

Hyperparameter optimization in our system is performed manually and focuses on stabilizing multimodal alignment while improving the richness and adequacy of generated text. For model training (Stages 1–3), we tune conventional parameters such as learning rate, batch size, weight decay, warmup ratio, and gradient accumulation steps, following the stability requirements reported in the Thought2Text paper. These manually selected parameters were sufficient for achieving stable convergence given the limited EEG dataset.

In addition to the training-time hyperparameters, we also optimized several inference-time hyperparameters, including the prompt template used to condition the language model. Our inference code implements three prompt types—basic, detailed, and creative—each corresponding to a predefined textual template that modifies how the model interprets the EEG-derived embedding. The basic prompt ("image labelstring Describe this image in one sentence:") produces short factual captions, whereas the detailed prompt encourages descriptions containing colors, background elements, and actions. The creative prompt, in contrast, elicits more expressive and imaginative descriptions. By systematically evaluating these prompt options during inference, we treated prompt selection as a tunable hyperparameter that directly affects caption richness and semantic adequacy. Incorporating prompt engineering into the hyperparameter optimization process allowed us to improve the descriptive quality of the generated text without modifying the underlying model weights.

This inclusion of prompt engineering as part of the hyperparameter search allowed us to meet the goal of improving output quality while keeping the model architecture unchanged. The prompt variations represent a lightweight yet highly impactful form of parameter optimization, aligning with modern LLM practices where prompt selection can function as a controllable inference hyperparameter influencing generation performance.

2.4 Scientific References Supporting the Methods

The methods in this work are directly supported by the design principles presented in the Thought2Text paper. The paper demonstrates that EEG signals are too noisy for end-to-end decoding and therefore require a three-stage training pipeline, including EEG–CLIP visual alignment, image-conditioned LLM fine-tuning, and EEG-conditioned projection learning. It further shows that grounding EEG in CLIP's semantic embedding space stabilizes training and significantly improves text generation quality. The decision to freeze the LLM and train only a lightweight projection layer is also justified in the paper, which reports that this strategy avoids overfitting and enables the LLM to interpret multimodal inputs effectively. Lastly, the use of GPT-4–generated captions and instruction-style text for supervision supports our use of enhanced prompting, as the paper highlights the importance of rich linguistic targets for producing coherent EEG-based text generation.

3. Evaluation

Evaluation of the EEG-to-Text generation system employed multiple complementary metrics to assess different aspects of performance. String-based metrics including BLEU, ROUGE (with variants ROUGE-1, ROUGE-2, and ROUGE-L), and METEOR measured surface-level similarity between generated and reference texts. Semantic metrics, particularly BERTScore, evaluated deeper meaning preservation by comparing contextual embeddings. We follow the standard evaluation methods for our results.

We got three csv results in total. In the csv files content 5 contents, which are Ground Truth Image, Expected objects, Predicted object, Expected Caption, and Generated Caption. The evaluation is mainly about the Expected Caption and Generated Caption. Using Jupyter notebook to finish the evaluation and get the data visualization. The results are as below:

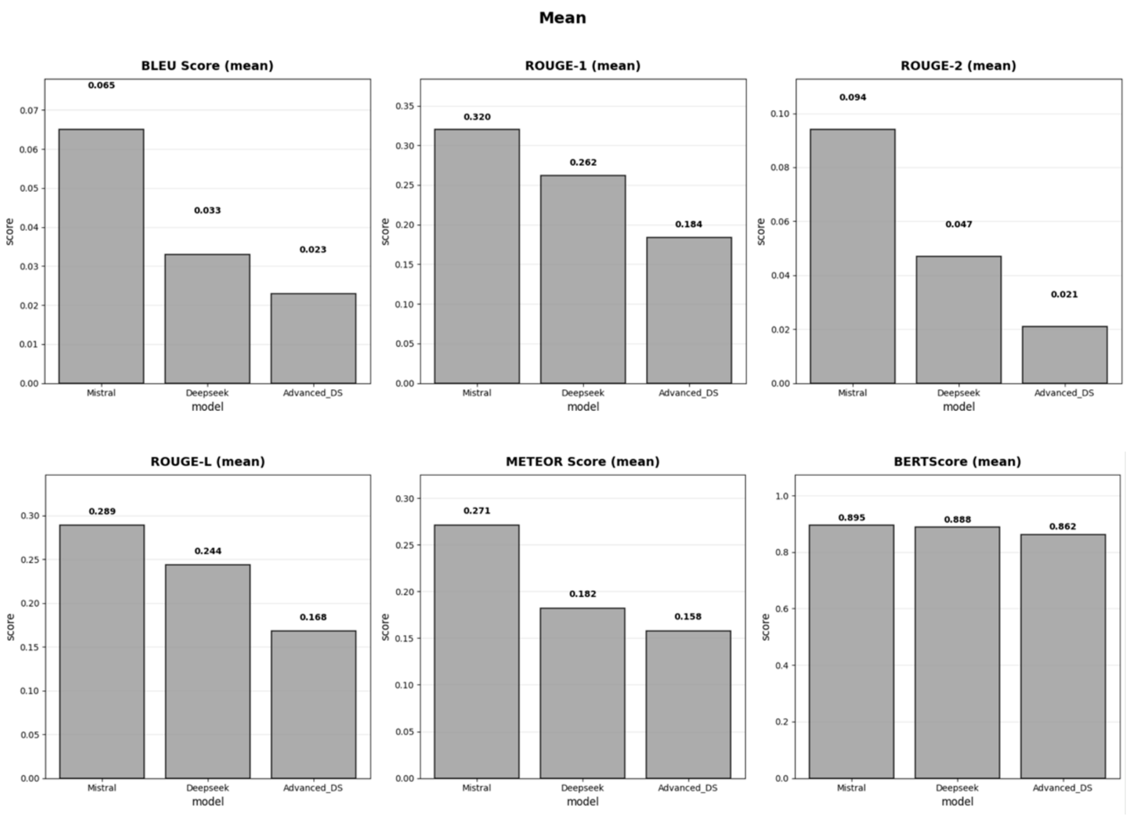


Figure 2: Evaluation results

The DeepSeek-7B-Base model demonstrated lower scores compared to Mistral-7B-Instruct on metrics evaluating exact word overlap such as BLEU, METEOR and ROUGE. And the Advanced inference DeepSeek model even got lower scores compared to other two models. However, it achieved nearly equivalent performance on BERTScore, which means all the model performance good in deeper meaning preservation.

Basic prompts with both models showed compact but metric-favorable performance profiles, while detailed prompts expanded the richness dimensions at some cost to traditional accuracy metrics. Comparative bar charts highlighted how different prompt types affected specific aspects like word count, vocabulary diversity, and descriptive term frequency. The qualitative results bad do not mean that the model performance bad. Qualitative examination of generated text provided insights beyond quantitative metrics. Example comparisons showed that while Mistral often produced more factually accurate descriptions matching expected objects, DeepSeek with detailed prompts generated more elaborate and contextually rich text, even when containing some inaccuracies. For example, where a mistal prompt gave ”A young girl is wearing a pink dress and sitting on a swing.” basic DeepSeek prompts produced ”A girl is sitting on a chair.”, and detailed DeepSeek prompts generated ”A young woman with long, dark hair is sitting on a couch. She has her legs crossed and she is looking at the camera." The downside is that in trying to be more descriptive, it sometimes made things up or got details wrong. There's a basic trade-off here between being detailed and being accurate.

The models' errors typically fell into three categories: misidentifying objects, leaving out details, or inventing elements that weren't there. The error also shows up during other LLM use cases. These patterns show specific opportunities for improving both the models and how users prompt them.

4. Conclusions

This project identified several key findings for EEG-to-Text generation. First, the choice of language model is crucial, as models differ in their ability to follow instructions and generate accurate text. Second, how the model is prompted has a major effect: more detailed instructions lead to richer and more descriptive outputs. However, this increased descriptiveness is often accompanied by a slight decrease in strict factual accuracy, highlighting a trade-off between detail and precision.

These findings are subject to certain limitations, including potential constraints on generalizability from the dataset, the inherent challenge of measuring creative quality, and the focus on single sentences rather than extended text. In summary, this project shows that DeepSeek-7B is promising for EEG-to-Text tasks, especially when used with detailed prompts to create richer descriptions. However, Mistral-7B performs better when the priority is metric-based accuracy. The strong influence of prompt engineering highlights that how we instruct the model is as important as the model itself for brain-computer interfaces.