# A APPENDIX

## A.1 ADDITIONAL INFORMATION ABOUT DATASET

The specific statistics of the dataset are shown in Table 4.

Table 4: The Statistics of EEG-ImageNet Dataset.

| | #Categories | #Images | #Subjects | #EEG-image pairs | Datasize |
|---|---|---|---|---|---|
| EEG-ImageNet | 80 | 4000 | 16 | 87,850 | 20.85GB |

As shown in Listing 1, the EEG-ImageNet dataset storage format is provided after review. The dataset can be accessed through the cloud storage link available in our GitHub repository after review. Due to file size limitations on the cloud storage platform, we split the dataset of Stage 1 into two parts: "EEG-ImageNet_1.pth" and "EEG-ImageNet_2.pth". Users can choose to use only one of the parts based on their specific needs or device limitations. Demographic information is also provided at the file level.

```
{
    "dataset": [
        {
            "eeg_data": torch.tensor,
            "granularity": "coarse"/"fine",
            "subject": 15,
            "label": 'ne2106550',
            "image": 'n02106550_1410.JPEG',
            "stage": 30/20, (This attribute only appears in Stage 2)
        }, ...
    ],
    "labels": [
        "n02106662", ...
    ],
    "images": [
        "n02106662_13.JPEG", ...
    ]
}
```

Listing 1: EEG-ImageNet dataset format.

## A.2 APPARATUS

All the image stimuli are presented on a desktop computer that has a 27-inch monitor with a resolution of 2,560×1,440 pixels and a refresh rate of 60 Hz. Participants are required to use the keyboard to interact with the platform. EEG signals are captured and amplified using a Scan NuAmps Express system (Compumedics Ltd., VIC, Australia) and a 64-channel Quik-Cap (Compumedical NeuroScan). A laptop computer functions as a server to record EEG signals and triggers using Curry8 software. Throughout the experiment, electrode-scalp impedance is maintained under 50Ω, and the sampling rate is set at 1,000Hz.

## A.3 EXPERIMENTAL SETUP DETAILS

We conduct experiments under three different granularity settings: the "all" task includes all 80 categories; the "coarse" task includes 40 coarse-grained categories; and the "fine" task includes 8 fine-grained categories that belong to the same parent node, with the average accuracy calculated across 5 groups.

The model structures and hyperparameters are as follows. For SVM, we try linear, polynomial, and radial basis function (RBF) kernels. The regularization parameter is tested from values $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}\}$. For RandomForest, we try to set the number of trees in the

forest from values $\{20, 50, 100, 200, 500\}$, with all other parameters set to their default values. For KNN, we set the number of neighbors to $\{5, 10, 15, 20\}$. For ridge regression, all parameters are set to their default values. For RGNN, when calculating the edge weights between electrodes, we use the hardware parameters of our data collection device to determine the topological coordinates of each electrode. In addition to the standard implementation, we add two batch normalization layers. The main hyperparameters adjusted are the number of output channels of the graph convolutional network (i.e., the hidden layer dimension) and the number of hops (i.e., the number of layers). These are set to $\{100, 200, 400\}$ and $\{1, 2, 4\}$ respectively. For EEGNet, we use the standard implementation and set the length of the first step convolution kernel to half the number of sampling time points, which is 200. The main hyperparameters adjusted are the number of output channels for the first convolutional layer (F1) and the depth multiplier (D), which are set to $\{8, 16, 32\}$ and $\{2, 4, 8\}$ respectively. For MLP, we set two hidden layers with dimensions of 256 and 128, respectively. Each linear layer is followed by a batch normalization layer and a dropout layer with a probability of 0.5. In the PT task, during the pre-training phase we train with half of the learning rate for half of the epochs.

For all deep models, we use the cross-entropy loss function. In MLP and EEGNet, we use the SGD optimizer with learning rate $10^{-3}$, weight decay $10^{-3}$, and momentum 0.9, training for 2000 epochs. After that, we adjust the learning rate to $10^{-4}$ and weight decay to $10^{-4}$ and continue training for another 1000 epochs. In RGNN, we use the Adam optimizer with learning rate $10^{-3}$ and weight decay $10^{-3}$, training for 2000 epochs. Subsequently, we adjust the learning rate to $10^{-4}$ and weight decay to $10^{-4}$ and train for an additional 1000 epochs. The batch size is uniformly set to 80.

All the implementations mentioned above are open-sourced and available in the GitHub repository.

### A.4 Additional Experimental Results

Table 5 shows the average results of all participants in Stage 1 and Table 6 shows the average results of all participants in Stage 2.

Table 5: The average results of all participants in the object classification task of Stage 1. * indicates the use of time-domain features, otherwise the use of frequency-domain features. † indicates that the difference compared to the best-performing model is significant with p-value $< 0.05$.

| Model | | Acc (all) | Acc (coarse) | Acc (fine) | F1 (all) | F1 (coarse) | F1 (fine) |
|---|---|---|---|---|---|---|---|
| Classic model | Ridge | 0.286±0.074† | 0.394±0.081† | 0.583±0.074† | 0.261±0.070† | 0.373±0.082† | 0.610±0.121† |
| | KNN | 0.304±0.086† | 0.401±0.097† | 0.696±0.068† | 0.286±0.081† | 0.380±0.096† | 0.717±0.132† |
| | RandomForest | 0.349±0.087† | 0.454±0.105† | 0.729±0.072† | 0.323±0.083† | 0.425±0.099† | 0.723±0.092† |
| | SVM | **0.392±0.086†** | **0.506±0.099†** | **0.778±0.054†** | **0.378±0.083†** | **0.486±0.105†** | **0.770±0.054†** |
| Deep model | MLP | 0.404±0.103† | **0.534±0.115** | **0.816±0.054** | 0.397±0.100† | **0.523±0.108** | **0.819±0.053** |
| | EEGNet* | 0.260±0.098† | 0.303±0.108† | 0.365±0.095† | 0.251±0.095† | 0.291±0.098† | 0.374±0.102† |
| | RGNN | **0.405±0.095** | 0.470±0.092† | 0.706±0.073† | **0.401±0.098** | 0.455±0.087† | 0.723±0.079† |

Table 6: The average results of all participants in the object classification task of Stage 2. * indicates the use of time-domain features, the use of frequency-domain features. † indicates that the difference compared to the best-performing model is significant with p-value $< 0.05$.

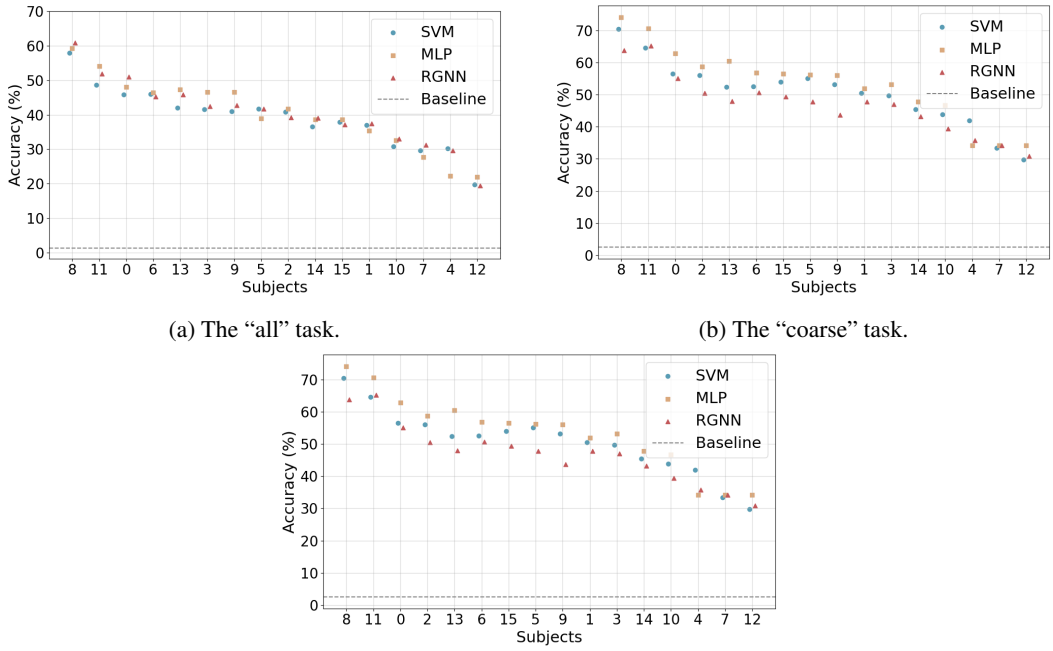
| Model | | Acc (all) | Acc (coarse) | Acc (fine) | F1 (all) | F1 (coarse) | F1 (fine) |
|---|---|---|---|---|---|---|---|
| Classic model | Ridge | 0.182±0.053† | 0.253±0.074† | 0.431±0.108† | 0.178±0.052† | 0.243±0.075† | 0.438±0.107† |
| | KNN | 0.220±0.081† | 0.310±0.113† | 0.574±0.119† | 0.211±0.083† | 0.299±0.105† | 0.565±0.134† |
| | RandomForest | 0.268±0.101† | 0.358±0.129† | 0.609±0.136† | 0.259±0.098† | 0.341±0.117† | 0.596±0.139† |
| | SVM | 0.281±0.090† | 0.368±0.107† | 0.657±0.140† | 0.271±0.084† | 0.365±0.109† | 0.648±0.134† |
| Deep model | MLP | 0.297±0.093 | 0.395±0.110 | **0.718±0.149** | 0.285±0.087 | **0.392±0.108** | **0.710±0.140** |
| | EEGNet* | 0.169±0.044† | 0.244±0.095† | 0.377±0.096† | 0.160±0.041† | 0.228±0.088† | 0.372±0.096† |
| | RGNN | **0.302±0.097** | **0.401±0.105** | 0.693±0.140† | **0.297±0.100** | 0.388±0.106 | 0.701±0.142† |

Table 7 shows the performance of the best-performing participant across all models and tasks of Stage 1..

Figure 5 shows the accuracy for each participant in the object classification task of Stage 1 across SVM, MLP, and RGNN models. We find that the ranking of participants' accuracy is relatively consistent across different models.

Table 7: The best results of all participants in the object classification task of Stage 1.

| Model | | Acc (all) | Acc (coarse) | Acc (fine) |
|---|---|---|---|---|
| Classic model | Ridge | 0.4550 | 0.5375 | 0.7200 |
| | KNN | 0.5025 | 0.6063 | 0.8013 |
| | RandomForest | 0.5006 | 0.6488 | 0.8450 |
| | SVM | **0.5794** | **0.7038** | **0.8588** |
| Deep model | RGNN | **0.6088** | 0.6525 | 0.8050 |
| | EEGNet* | 0.4413 | 0.5213 | 0.5988 |
| | MLP | 0.5925 | **0.7413** | **0.8875** |




(a) The "all" task.

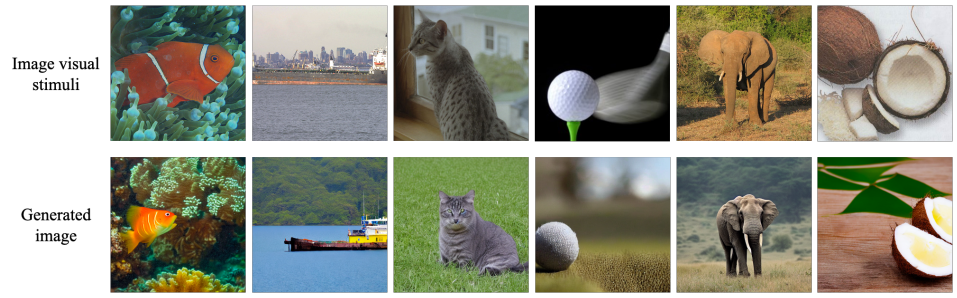
(b) The "coarse" task.

(c) The "fine" task.

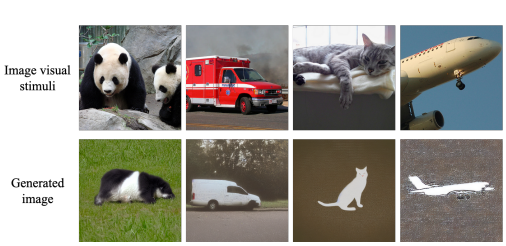
Figure 5: Acc for each participant in the object classification task of Stage 1 across SVM, MLP, and RGNN Models.

Figure 6 presents more image generation results selected from other participants, with Figure 6a showing good cases and Figure 6b showing bad cases. We identified **three** main types of bad cases. Similar to the first two images, the reconstructed images lack or misrepresent low-level information such as color and shape. These errors are relatively common and are due to the limitations of our feature mapper and the simple structure of the reconstruction pipeline, resulting in insufficient information restoration. Similar to the latter two images, the reconstructed images lack detail. This limitation is due to the number of denoising steps in the diffusion model and the inherently low signal-to-noise ratio of EEG signals.

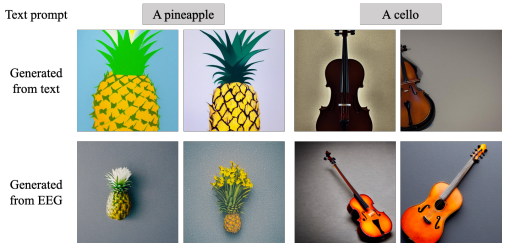
We also observed that for certain categories, especially fine-grained ones, all test data points resulted in near-noise outputs, which drew our attention. When we directly input category labels as text prompts into Stable Diffusion 1.4, we found that the generated images had poor realism and three-dimensional structure. Figure 6c compares these images with those generated by our reconstruction pipeline from the training set. This improvement suggests that we can use EEG, which can be quickly and extensively obtained as human feedback signals, to enhance the performance of text-image pre-trained models or generative models. This will be the direction of future research.




(a) Good cases.




(b) Bad cases.

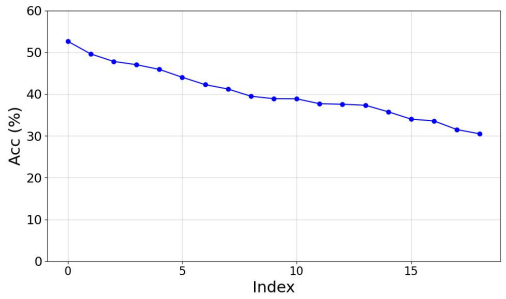



(c) Comparison of EEG and text-generated images.

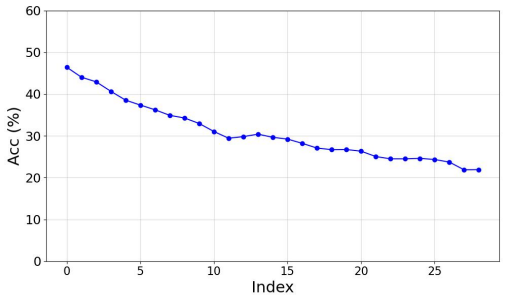
Figure 6: More results in the image generation task.

### A.5 Temporal Effect

In Figure 7, we plotted the average classification accuracy for images at different index positions in the test set under various training and test set splits to show the temporal effect in Stage 1.




(a) 30/20 split for the training and test sets.




(b) 20/30 split for the training and test sets.

Figure 7: Average classification accuracy under different training and test set splits, with accuracy plotted against the indices of image stimuli in the test set.

We observed that the first few images in the test set have significantly higher accuracy, indicating a strong temporal effect.

### A.6 The Use of Large Language Models

In this work, we leveraged large language models (LLMs) to assist in manuscript preparation, including refining the text for clarity and style, as well as facilitating literature retrieval. All LLM-generated suggestions were carefully reviewed, edited, and integrated by the authors to ensure scientific accuracy and consistency with our own writing voice. We acknowledge the ongoing discourse around the ethical use of LLMs in scholarly writing—particularly regarding transparency, originality, and accountability. We transparently report the use of LLM assistance and reaffirm that all substantive intellectual contributions (e.g. experimental design, data analysis, interpretation) originated from the authors.