EEG-Vision: Deep Learning for Generalized EEG-based Image Visualization

Stefan Baumann
Carnegie Mellon University
sbaumann@andrew.cmu.edu

Ammar Karkour
Carnegie Mellon University
akarkour@andrew.cmu.edu

Sideeg Hassan
Carnegie Mellon University
sahassan@andrew.cmu.edu

EuiSuh Jeong
Carnegie Mellon University
esj1@cmu.edu

Abstract

The human brain is one of the biggest mysteries that scientists have been studying for centuries. A big part of this research is understanding how different types of sensory input affect the brain's activity. Historically, studying visual perception had been done through brain anatomy and cognitive psychology, but recently the rapid increase in the amount of publicly available neural data has provided the opportunity for researchers without knowledge of neuroscience to use deep learning techniques to simulate and reconstruct how our brains perceive visual inputs. For instance, classifying EEG signals when viewing certain images to their corresponding image classes. Several attempts were made to solve this problem, such as PeRCeiVe Lab who used a convolutional neural network that resulted with a testing accuracy of 48.1% on this classification task. In this project, we improve on their work by creating **EEG-Net** - a recurrent neural network (i.e. a BiLSTM model) - that achieves a testing accuracy of 53% on classification of PeRCeiVe Lab's dataset. Moreover, we use a pre-trained **Image-Net** model to extract image embeddings, which are used along with our EEG embeddings to train a third model, which we call **EEG-Image-Map**. This mapping model takes EEG embeddings as an input and returns an image embedding as output in an attempt to visualize images, whose classes were not part of the training set similar to zero-shot learning.

1 Introduction

The aim of this project is to determine what an individual is seeing using their brain signals. This is an important task because it will give insight into how visual perception affects brain activity. Since we will be using non-invasive EEG data to represent brain activity, we will also gain insight into the possibility of inferring visual information from EEG signals alone, which is partially unknown until now.

Previous research has shown that EEG signals can be used for image classification, i.e. using a model to infer what class of image a subject is viewing from a fairly small set of image classes used in training. Our study attempts to build on this by creating a system composed of multiple models to infer image features of image a subject is viewing from a larger set of image classes that have **not** been used in training the model that processes the subject's brain signals. If successful, this means that our system could be scaled up to be able to infer a viewed image from any set of image classes no matter how finitely large the set is. Though, we have reason to believe that the system could also infer possibly infinite classes of images (i.e. generate the image being viewed from the corresponding EEG signals alone), however this is not the main goal of the project.

Our system mainly consists of three models. The first model processes raw EEG data and is used to extract features of the EEG which intuitively corresponds to the image that the subject was viewing when recording the EEG signals. The second model processes an image and is used to extract its respective image features. The EEG features of the first model is used along with the image features extracted from the second model to train a third model that generates image features from EEG features. Lastly, we use the features extracted from our third model to identify which image the subject was viewing from a testing set of image classes, which were not part of the training set.

2 Literature Review

During the period of rapid development of deep learning in the industry, numerous scientists utilized EEG data to train neural networks for fields such as health care, communication, and security. They were hoping that the analysis of this type of data could boost the performance of models aimed at solving problems where neural activity is a correlated or even causal variable. Compared to other physiological data, EEG directly showcases human cortical activity with very high temporal resolution and fair spatial resolution, while being non-invasive, which is an important factor when data collection should be as scaleable as possible because the most important part of a model is its training data. Interestingly, this type of research is appreciated in both fields of computer science and neuroscience because each model could essentially describe how our physical brains approach these tasks. The focus of this literature review is specifically about making inferences about images using the EEG data of a subject seeing those images and the images themselves, therefore we explore previous work done on this task and the use of Computer Vision models for this task respectively.

In general, the work that has been done in this area was centered around the goal of classification to a set of classes the models were trained on. For instance, training a model using a dataset of images of numerous classes and EEG signals representations of images, for the purpose of correctly classifying a given EEG signals into one of the image classes. We found three studies that specifically worked on this problem, and all of them used the same dataset collected by PeRCeiVe Lab. This dataset is comprised of EEG recordings of subjects viewing an image from a subset of 40 classes (originating from the ImageNet database [3]) - making it a 40 class EEG-based classification task.

The first paper came from the PeRCeiVe Lab. who presents three models that uses the EEG signals as inputs and outputs the respective image class [7]. Their best model's architecture consisted of stacked LSTMs and an additional output layer, which included a linear combination of the input and a ReLU. They specifically used the LSTM recurrent neural network because of its temporal dynamics, since the EEG input is a time-series. Their reported testing accuracy for the 40 class classification task was 82.9%, which outperformed their defined baseline by around 70%. As a baseline they used a statistical approach known as Linear Discriminant Analysis, specifically using the implementation presented by Kanesiro et al. [5]. Another insight they made was that they saw a slight (1%) improvement in testing accuracy when they used the EEG data from the time interval of 320-480ms rather than the complete time interval (after preprocessing) of 40-480ms.

The second paper was released by Fares et al. in 2019 [4]. They proposed an approach where they first extract region-level information and then passing the new data into their model. The model architecture they use is comprised of a stacked bidirectional LSTM network with an ICA and SVM method (instead of a typical SoftMax) for classifying the resulting embedding. With this approach they reach an outstanding 97.3% accuracy to classify an image that a subject has seen using their EEG signals. In their paper they conclude that this is the current state-of-the art for this task by comparing their accuracy with the three previously mentioned approaches.

The third paper was also presented by PeRCeiVe Lab. In this paper they presents a different approach to the same classification problem [6]. This time they present a system that uses a siamese network for learning joint-brain image representations. Their proposed siamese network uses an EEG encoder and an image encoder with which they try to maximize the similarity between the EEG features and image features. The EEG encoder is comprised of multiple 1D convolutions across temporal and spatial channels, then the resulting features are processed by multiple residual layers, a final convolution, and a fully-connected layer to output an embedding of 1000 features. The image encoder used is simply a pretrained instance of one of the popular ImageNet classification model (i.e. Inceptionv3 [8]). They reached a 48.1% accuracy by only using their EEG encoder as the classifier, but a 60.4% accuracy when applying their joint learning method. Surprisingly they state that this approach surpasses

Table 1: EEG-ImageNet datasets

Dataset	# subjects	# image classes	# images/class	# entries	# EEGs x samples
[7][6]	6	40	50	12000	128 x 500
[1]	1	14012	1	14012	5 x 384

any other previous approaches because they explain that in their 2019 study they carried out a data processing step (i.e. frequency filtering) incorrectly, which lead to the data in the used dataset being biased. In conclusion, the results of their first paper and the paper by Fares et al. (2019) cannot really be used for comparisons as they both used the biased dataset. This implies that PeRCeiVe Lab's results from their second paper is the current state of the art - 60.4% testing accuracy.

3 Datasets

The data needed for this task is EEG data from subjects viewing images, whose image is available and whose class is labeled. For example, we need an image of a tree and the EEG signals recorded of a subject viewing that image. We also wanted the image samples to come from the ImageNet database because part of our system is an image classifier trained on the ImageNet data. We found two such datasets that are publicly available as seen in Table 1.

We have decided to use the dataset by PeRCeiVe Lab [7][6] mainly because most of the related work surrounding the problem we wanted to solve used this dataset too. It consists of 500 EEG recordings corresponding to 40 image classes, each class having 50 individual images. The data was collected from 6 subjects being shown each image for 500ms. Their EEG data was gathered using 128 electrodes, recording at a frequency of 1000Hz.

The data collection and preprocessing happened as follows:

3.0.1 EEG Data

Collection

The EEG data coollection was quite straight forward because the link to the dataset[1] could be found on PeRCeiVe Lab's website[2]. The data comes as a *.pth* file, which means it is easily loadable using the *torch* Python module[3].

Preprocessing

The lab offers the dataset already filtered in three frequency ranges: 14-70Hz, 5-95Hz, and 55-95Hz. We opted for the 55-95Hz filtered data because that is what the current SOTA model used [6]. We split the data into training, validation, and testing using a 80%, 10%, 10% split respectively. Due to the fact that there are multiple EEG representations for each image (because all subjects saw the same images), we had to make sure that all EEG representations for a single image was included in a single split. This was done similarly by all the related work [7][6][4].

3.0.2 Image Data

Collection

Since the EEG dataset only provided us with the individual ImageNet image file names and not with the images themselves, we had to extract the images used in the EEG dataset from the original ImageNet dataset [3]. This was a challenging task because there is no way of downloading specific images using their file names, so we had to download the entire dataset, which takes a lot of memory. We downloaded the complete dataset on an AWS EC2 instance with 500GB of storage and

[1] `https://tinyurl.com/eeg-visual-classification`

[2] `http://www.perceivelab.com/index`

[3] `https://pytorch.org`

wrote a Python script to extract the images we needed to be used locally.

Preprocessing

The ImageNet images come as *Pillow*[4] Image objects so we were able to easily create a dataset with transforms from *torchvision*[5]. For the training dataset we used the following transforms: *RandomResizedCrop, RandomHorizontalFlip, ToTensor, Normalize*. For the validation and testing datasets we used the following transforms: *Resize, CenterCrop, ToTensor, Normalize*. The mean and standard deviations for the normalization were as follows: [0.485, 0.456, 0.406], [0.229, 0.224, 0.225].

All data loading and preprocessing can be found in the *Data_preparation_EEGVision* notebook in our Github repository here.

4 Model Descriptions

Looking at the previously mentioned approaches, we can see that they have shown promising results doing image classification using EEG data. However, they are still limited to predicting the number of classes that are part of the training set. On top of that, they cannot identify the exact image features the subject saw. This is precisely the motivation of our work. In order to fill this gap, we propose a system that can map any EEG signal to the image it represents, using multiple features extraction models and a features mapping procedure.

The goal of our project is to implement a feature mapping procedure that maps features extracted from the EEG signals to the actual images that the EEG signals represent. More specifically, we have 3 main models. The first model extracts features from the images themselves, and the second model extracts features from EEG signals that represent those images, we call these two models **Image-Net** and **EEG-Net** respectively. In addition, we have a third model **EEG-Image-Map**, that takes in the extracted EEG features and generates their respective image features, using extracted image features from Image-Net as labels during training. Look at figure 1 for an illustration of our system.

In light of previous papers' results, we believe that we have two aspects of novelty. Firstly, the model we use to extract the EEG features (EEG encoding) has a reported accuracy of 53% during **validation** on classification tasks compared to PeRCeiVe Lab second paper's EEG encoding model that reported 48% accuracy classification accuracy when only using EEG data for the task. Secondly, our system is scalable to more than the limited number of classes that are used to train the model for EEG feature extraction with help of a pretrained Computer Vision model, which to our knowledge was not done before.

If our mapping works well for the classes in the training set, then we **hypothesize** that the mapping should be able to generate accurate image features irrelevant of whether its class was in the training set or not, because we are generally learning how to predict image features from EEG features, which is irrelevant of what these EEG features represent. We want to harness this generality and use it to find the image class that an EEG signals represents when the class is not part of the training set.

4.1 EEG-Net

The purpose of EEG-Net is to encode a sequence of EEG signals recording into a vector of features, which is used later to generate a vector of image features. A similar approach was taken by Palazzo et al. (2020) in their last paper where they used an EEG encoder to encode EEG signals recording[6]. However, while they use a CNN based model for this task we use a bi-directional LSTM based model. Our model is incremental work that builds on Fares et al. (2019) work [4].

Since an EEG signals recording is a recording over time, then this implies that there is a temporal correlation between the different units of an input sequence. Based on that, we decided to build on Fares et al. (2019) work that uses a bi-directional LSTM.

[4] `https://pillow.readthedocs.io/en/stable/`

[5] `https://pytorch.org/vision/stable/transforms.html`

Our model uses CNN layer as an input layer, following that we use a 3 stacked bi-directional LSTM layers each one of them has 128 hidden nodes (similar to Fares et al. (2019)), and lastly, we have 2 fully connected layers, one for extracting 128 features and another one that after it to classify these 128 features to one of 40 classes. During training we use the last layer to classify and the layer before it to extract the EGG features.

4.2 Image-Net

As for our Image-Net baseline model, we chose the Inception_v3 model proposed by Szegedy et al. [8], because this model was reported by PeRCeiVe Lab to have the highest classification accuracy in one of their papers [6]. This model uses suitably factorized convolutions and aggressive regularization to scale up computer vision networks while making use of the added computation as efficiently as possible. The model achieves 21.2% top-1 and 5.6% top-5 error benchmarks on the ILSVRC 2012 classification challenge validation set, which included 1000 image classes.

For our purposes we are using a pretrained inception_v3 model where we modified the final layer during training to change the number of output classes to 40 in order to match the classes that we have available in our EEG dataset. Then we updated the weights of final layer by retraining only that layer on the training dataset with only the 40 classes. In comparison to PeRCeiVe Lab's implementation, which reached a 93.1% accuracy, we are doing similarly well with a 98% testing accuracy, which is understandable since we are only classifying to 40 classes compared to the 1000 classes they do. For feature extraction we use the outputs of the second-to-last layer, which is comprised of 512 neurons to produce a image feature embedding of dimension 512 that will be used to train our EEG-Image-Map.

4.3 EEG-Image-Map

The goal of EEG-Image-Map is to predict a vector of image features using vector of EEG features. As descried earlier, to train this model we pass EEG signals recording to EEG-Net to extract EEG features and use them as **data**, and we also pass the actual images that these EEG signals represent to Image-Net to extract image features and use them as **labels**. Following that our model learns to predicts an image features vector of the same size as the labels vectors based on the EEG input features vector.

Since our inputs are just features, meaning they don't have any temporal nor spatial correlation, we used a multi-layer perceptron (MLP) as our architecture, . Our MLP has 5 layers of sizes 128, 256, 256, 512, 512. For training this model, we use a **cosine similarity** based loss which calculates how similar are the output vector and the label vector, with the goal of making them as close as possible. We hypothesize that the closer the output features vector to the actual vector, the better we can classify that vector or exactly identify it.

Notice that since our system generates image features from EEG features, then another generative model could be used to generate the actual image using the generated image features. However, since reaching this goal depends mainly on the performance of our system in generating accurate image features, this project just focuses on generating accurate image features.

5 Evaluation Metrics

Since we use three separate models to create our prediction system, we split the evaluation metric accordingly.

5.0.1 EEG-Net

We evaluate our EEG-Net performance by measuring the **accuracy** of our model in classification tasks on our test set. More specifically, following the feature extraction, we classify these features into one of the image classes we have EEG data for, and monitor how accurate the classification is. Since our overall goal is to use this model for EEG feature extractions, then the higher the classification accuracy is, the more representative the feature extraction will be.

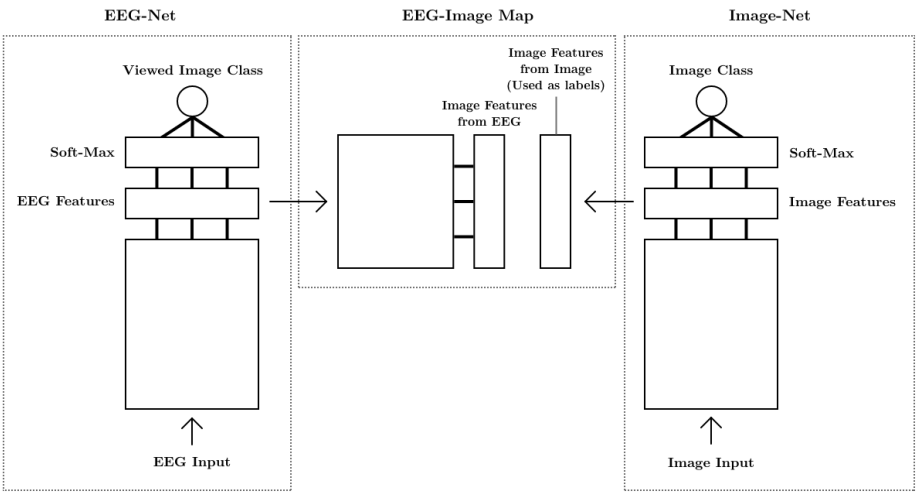

Figure 1: Illustration of proposed system comprised of EEG-Net, Image-Net, and EEG-Image Map.

5.0.2 Image-Net

We evaluate our Image-Net model also by measuring its accuracy in classification tasks on our test set. This test set includes the same 40 image classes used in the EEG-Net model. We measure our accuracy by calculating the percentage of correctly classified test instances from the test set. Here again, higher accuracy should indicate better feature extraction, which is important for our EEG-Image-Map.

5.0.3 EEG-Image-Map

When predicting the image features from EEG features, we use **two** evaluation metrics.

For the first metric, we first pass the actual image that the EEG signal represents to our Image-Net to get its actual features (notice that Image-Net has very high accuracy, meaning the image features extracted from it are very accurate and descriptive). After that, we get the generated image features from our EEG-Image-Map. Lastly, we use **cosine similarity** to measure how close the generated features are to the actual features. We chose this evaluation metric as we hypothesize that the better the feature generation is, the **closer** the generated features will be to the actual image features.

For the second and third metric, we find the image feature from the test set whose cosine similarity is highest with the generated image feature. Then we check if the selected image is part of the correct image class and if the image was the actual image, we call these two metrics **classification accuracy** and **identification accuracy**. We chose this evaluation metric as we hypothesize that the better the generated features are (i.e. the closer to the actual image features they are), the higher the classification/identification accuracies will be.

Notice that these three evaluation metrics are used to measure the performance of our model on classes the model was trained on and on classes the model wasn't trained on (to measure how well our model scales to unseen classes). To do so, we perform an **out-of-training verification** task. In this task we test if the system can predict the image class that the EEG-Net was not trained on. One way to do this is to split our data so that only 35 image classes (out of 40) are used for EEG-Net training and validation. For testing we pass new EEG signals of the 5 left-over image classes that were not used for EEG-Net training into the EEG-Net model and then their EEG features into the EEG-Image-Map to generate EEG-based image features. Following that, we use both of our evaluation metrics to measure the performance of our model on these newly generated features.

6 Baseline Models

6.1 EEG-Net

As a baseline model for EEG-Net, we chose the aforementioned model by Fares et al. [4]. Even though it was trained on biased data, it gave better results than PeRCeiVe Lab's first paper [7], which was also trained on biased data. So, we hypothesised that the architecture might give us better eeg encodings that the eeg encoding model that was used by PeRCeiVe Lab. That model uses bi-directional LSTM to capture hidden dynamic correlations in the EEG data. We implemented this model based on the model description on the paper.

6.2 Image-Net

As for our Image-Net baseline model, we chose the Inception_v3 model proposed by Szegedy et al. [8], because this model was reported by PeRCeiVe Lab to have the highest classification accuracy in one of their papers [6]. This model uses suitably factorized convolutions and aggressive regularization to scale up computer vision networks while making use of the added computation as efficiently as possible. The model achieves 21.2% top-1 and 5.6% top-5 error benchmarks on the ILSVRC 2012 classification challenge validation set. An open source code was used for this model except the part where we fine-tune the model to make it match our purposes.

6.3 EEG-Image-Map

A multi-layer perceptron (MLP), which is a type of feed forward artificial neural network, was used to build a model that maps EEG features to image features that is retrieved from EEG-Net and Image-Net respectively. This model was implemented from scratch by us.

Our implementation can be found on our GitHub repository **here**.

7 Experiments

Perfecting our system mainly involved training three individual components, each component had its own experiments that we discuss here. We kept track of all the model performances using a platform called Weights and Biases [2]. The data to all the experiments can also be found on our GitHub repository.

7.0.1 EEG-Net

The main challenge in this part was figuring out the parameters that were used to implement this model. That's because in their paper, Fares et al. (2019) just mention the kind of layers they use without mentioning most of the parameters of these layers. Moreover, they didn't talk at all about any optimization method they used.

We tried first to implement the layers without any optimization methods, but we realized that the model over-fits and it doesn't learn anything after reaching 35% accuracy. So, we added a dropout after each BiLSTM layer of probability of 0.4 and we used a scheduler that delayed the over-fitting a little bit and helped in accelerating the learning process. With these 2 optimization methods we were able to reach an accuracy of 47%.

Moreover, since the EEG recordings are recorded using different channels placed on different location on the subject's head, we thought that we can use a CNN layer as an input layer to capture the spatial correlation between units of each recording. Doing so, boosted our classification accuracy from 47% to 53% which is better than Palazzo et al. [2020]'s EEG encoding model [6]. We believe that we reached better results because we capture both temporal and spatial correlation.

All in all, we tuned our model with the following hyperparameters: dropout rate, learning rate, number of hidden nodes.

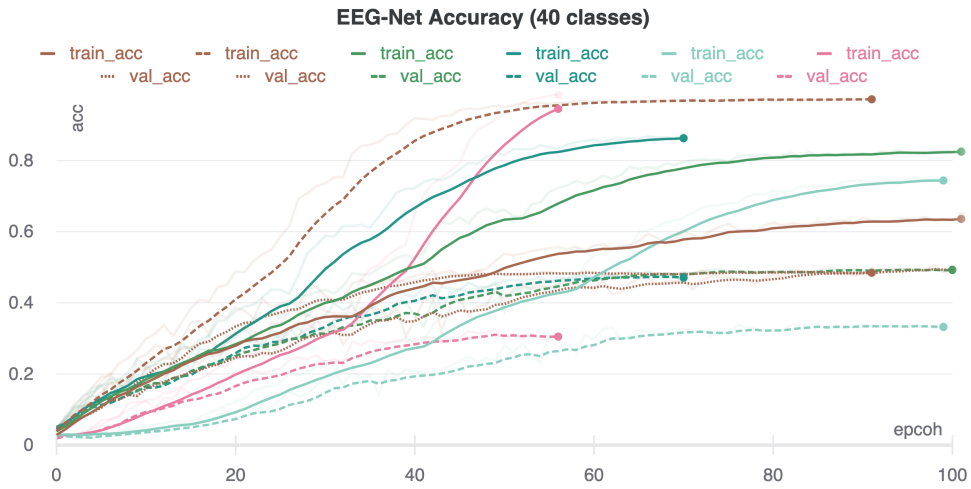

Figure 2: Accuracy curves for different EEG-Net configurations we tried out.

We also implemented the model proposed by Spampinato et al. (2019) but this model usually over-fit very quickly and in the end, it never outperformed the one by Fares et al. We used CrossEntropy loss for all model configurations. See figure 2 to see our most diverse models performed during training.

7.0.2 Image-Net

We fine tune our pretrained Image-Net by changes the last layers size to be 512 (required extracted features), and we also add another layer after it of size 40 that classify the extracted features to one of 40 classes.

After that we retrain the last 2 layers to make the model adjust to our dataset. Finally, we remove the last layer and use Image-Net to extract image features.

7.0.3 EEG-Image-Map

To implement this model we experimented on 3 different model, simple MLP, BiLSTM, and MLP. We also experimented with 2 loss functions, nn.MSELoss() and nn.CosineEmbeddingLoss() which is a Pytorch cosine similarity based loss function.

The choice of the loss function was based on the fact that we want a loss function that compares between 2 fixed size vectors. We also needed a loss function that matches our evaluation metric. While MSELoss satisfies the first condition, it doesn't match out evaluation metric as well as CosineEmbeddingLoss does, because our evaluation metric is based on measuring how similar 2 vectors are using cosine similarity. Therefore, we decided to use CosineEmbeddingLoss as our loss function.

Following that we tested our different models. The simple MLP consisted of 3 layer (input, hidden, output). This model didn't perform really well as it was very simple and it didn't learn from training. Following that we tested with the BiLSTM, however, it also didn't perform well as expected as there is no any temporal correlation between the extracted features. Lastly we used a bigger MLP with 5 layers, each layer is followed by a BatchNorm, a ReLU and a dropout of probability 0.2. Using this architecture and these optimization methods, we succeeded in reaching high cosine similarity on in-training images and out-of-training images.

All in all, we tuned our model with the following hyperparameters: EEG and Image feature sizes, MLP layers, dropout rate. See figure 3 to see our most diverse models performed during training. In figure 3 we see two groups of models, the group of models that did not perform better than 0.4 loss are the ones with EEG feature and Image feature dimensions of 60 and 40 respectively. The other models used dimensions 128 and 512 respectively.

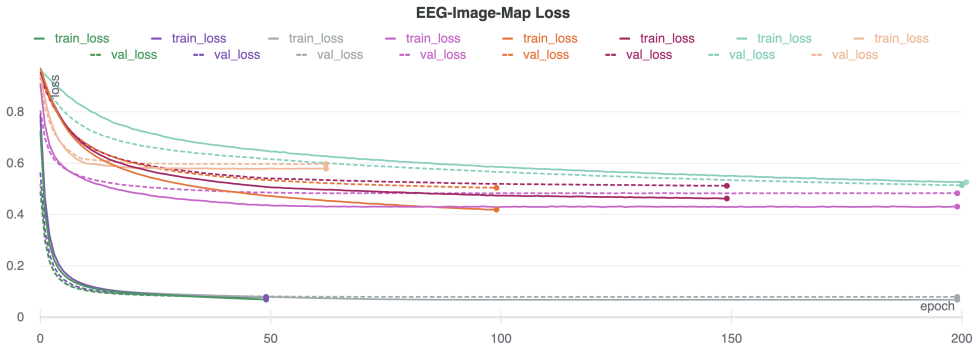

Figure 3: Loss curves for different EEG-Image-Map configurations we tried out.

```
EEG-Net test  accuracy: 47.9429 %
Image-Net test  accuracy: 99.4975 %
EEG-Image-Map avg test cosine similarity: 0.9178
EEG-Image-Map test classification accuracy: 46.683459 %
EEG-Image-Map test identification accuracy: 1.511335 %
```

Figure 4: Results for phase 1 testing.

8 Results and Discussion

We discuss the results in the context of two phases. Our best EEG-Net model was the Fares et al. architecture with the following specifications: 256 hidden layers, 128 number features, 0.001 learning rate, 0.4 dropout rate. Our best EEG-Image-Net model was the MLP architecture with the following specifications: [128, 256, 256, 512, 512] linear layers, 0.1 dropout rate, 0.00001 learning rate. Our Image-Net model extracted image features of size 512.

8.1 Phase 1

In the first phase we train and test all the models in our system with all 40 image classes. The system is then evaluated using the metrics as described above. As seen in figure 4, our EEG-Net reached 47.9% testing classification accuracy and 53% validation classification accuracy, our Image-Net reached 99.4% testing classification accuracy on all 40 classes. Our average testing cosine similarity was 0.9178, which we are happy about as the closer to 1 it is, the more similar the generated features are to the true features. Interestingly, our classification and identification accuracies for EEG-Image-Map are 46.6% and 1.5% respectively, which means that our mapping model does not significantly reduce the probability of correct classification. The 1.5% is not bad considering that we have around 1200 images in the testing set, so it is around 18 times better than randomly guessing the image.

8.2 Phase 2

During phase 2 we assume that we have a large image dataset but no EEG signals for it yet, but when we get an EEG signal we want to be able to predict what image and image class it represents, even if it is not part of the training dataset for the EEG-based models.
First, we selected 5 image classes out of the total 40 image classes to use for the testing set. We created the training and validation datasets using the other 35 classes. We retrain the EEG-Net and EEG-Image-Map because they should only have seen EEG data from the 35 image classes selected for training and validation. We train the Image-Net with all 40 image classes because we are assuming that we have a large image dataset already available. Next, we create an array of image features by extracting the features from all of the testing set data by inputting it into Image-Net. Now, we can pass the testing data through our EEG-Net and EEG-Image-Map to evaluate it using the metrics we described previously.

```
EEG-Net val accuracy: 52.7778 %
Image-Net test accuracy: 99.6 %
EEG-Image-Map avg test cosine similarity: 0.8652
EEG-Image-Map test classification accuracy: 20.066667 %
EEG-Image-Map test identification accuracy: 0.066667 %
```

Figure 5: Results for phase 2 testing.

```
EEG-Net val accuracy: 52.7778 %
Image-Net val accuracy: 98.2759 %
EEG-Image-Map avg val cosine similarity: 0.9217
EEG-Image-Map val classification accuracy: 44.636015 %
EEG-Image-Map val identification accuracy: 1.245211 %
```

Figure 6: Results for phase 2 validation.

For phase 2, we want to mainly focus on the testing average cosine similarity. Our EEG-Image-Map reached **0.8652**, which we are pleased about because it means that our mapping model is able to **generalize** well and learn and generate feature mappings that are still very similar to the true feature embedding, although the classes were never shown to the EEG-Net and EEG-Image-Map, **which is to our knowledge was not done before**. Interestingly, our classification and identification accuracies for EEG-Image-Map are 20% and 0.06% respectively, which means that our mapping model does not do better than simply guessing the correct class and image from the testing dataset (as this set only contains 5 image classes). This is unfortunate because maximizing feature similarity did not correspond to better classification and identification accuracies, which was our null hypothesis. We believe that this is mainly because the aim of the chosen loss function was to maximize cosine similarity not classifying or identifying the features, however, this issue will be studied more in the future work.

9 Conclusion

In this project, we aimed to solve the problem of visualizing EEG signals of visual stimulus, whether they were part of our training dataset **or not**. To do so, we implemented a system that predicts image features based on the input EEG signals recording. Following that, we evaluate the performance of our system using three metrics. The first one is by measuring how close the predicted image features compared to the actual features of the image, by calculating the **cosine similarity** between them. For the second and third metric, we find the image features from the test set whose cosine similarity is highest with the generated image features. Then we check if the selected image is part of the correct image class and if the image was the actual image.

Our system consists of three main models. The first one is a bi-directional LSTM based model that returns EEG features from EEG signals recording. The second model is an Inception_v3 model that returns image features from the actual images. The last model is an MLP that returns a vector of image features from a vector of EEG features, and uses the outputs of the first two models as data and labels during training.

Our system proved to generate image features that are very similar to the original image features, with cosine similarity of 0.9217 for image classes that were in the training data set and 0.8652 for image classes that were not in the training dataset, which means that our system **generalizes** very well to unseen classes, which is to our knowledge was not done before (figures .3 and .4). However, our system didn't do well in classifying these image features and exactly identifying them, and we believe that this happens because we didn't train our model to do that task, as the main goal for our model was to generate similar image features, not to classify them or identify them.

10 Future Work

As a future work we aim to implement a better EEG features extraction and classification model. As mentioned before the current state of the art for classifying EEG signals is presented by Palazzo et al.

[2020], where they use an EEG features encoding model and combine it with image features to build a joint learning based model reaching a classification accuracy of 60.4% [6]. We believe that we can use our EEG features encoding system that reached almost 53% validation and testing classification accuracy (figures .3 and .4) compared to the 48% that their EEG encoding model reach, and combine it with their join learning technique to reach a new state of the art in classifying EEG signals to a limited number of classes.

Moreover, since our system proved to reach a very high cosine similarity between the generated features and the actual features, we believe that we can use a generative model to generate images that are very similar to the original images with an accuracy that was not reached before.

Lastly, we are planning to study the problem of classifying and identifying images that were not in the classes of the training in more depth and aim to solve it using a forth model that is trained to classify our very good extracted image features.

11 Division of Work

The tasks of the research were divided into mainly constructing different sector of the system: Image-Net (Model responsible for extracting image features from images), EEG-Net (Model that extracts EEG features from EEG signals), and EEG-Image-Map (Model that maps EEG features to Image features), as well as researching and documenting our work.

Searching for the gap and the research question was done by all of us, while detailed literature review and setting up the datasets was mainly done by Stefan. Implementing the EEG-Net was mainly done by Ammar with the help of Stefan. While, setting the Image-Net and fine tune it was mainly done by Sideeg and Stefan. Following that, implementing EEG-Image-Map was mainly done by Ammar and John with the help of Stefan. Lastly, implementing the pipeline and the tracking system (Weights & Biases) was mainly done by Stefan with the help of Sideeg and Ammar, as well as experimenting with different parameters and architectures.

Eventually all the models had to integrate into a system, thus we have held two meetings every week to report on the progress and different information that needs to be shared. This was done in order to prevent any conflict that could happen in the process of integrating different models. Research and literature review were done in groups. These groups were divided according to which part of the system that person was responsible for. For instance, information like relationship of EEG signals and image recognition was studied together as it was required to know regardless of which model construction that member is working on.

References

[1] URL: http://www.mindbigdata.com/opendb/imagenet.html.

[2] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[3] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[4] Ahmed Fares, Sheng-hua Zhong, and Jianmin Jiang. "EEG-based image classification via a region-level stacked bi-directional deep learning framework". In: *BMC medical informatics and decision making* 19.6 (2019), pp. 1–11.

[5] Blair Kaneshiro et al. "A Representational Similarity Analysis of the Dynamics of Object Processing Using Single-Trial EEG Classification". In: *PLOS ONE* 10.8 (Aug. 2015), pp. 1–27. DOI: 10.1371/journal.pone.0135697. URL: https://doi.org/10.1371/journal.pone.0135697.

[6] Simone Palazzo et al. *Decoding Brain Representations by Multimodal Learning of Neural Activity and Visual Features*. 2020. arXiv: 1810.10974 [cs.CV].

[7] Concetto Spampinato et al. *Deep Learning Human Mind for Automated Visual Classification*. 2019. arXiv: 1609.00344 [cs.CV].

[8] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.