Università degli Studi di Milano Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Exploration and comparison of deep learning architectures to predict brain response to realistic pictures

Relatore: *Dimitri Ognibene*

Co-relatore: *Giuseppe Vizzari*

Relazione della prova finale di:

Riccardo Chimisso

Matricola 866009

Anno Accademico 2022 - 2023

Table of contents

Special thanks

Apart from the professors that accompanied me during this remarkable experience, I want to thank Paolo Marocco and Sathya Bursic, who closely followed me during development, helping me face and deepen many new concepts and code-related matters. I also want to thank several friends and colleagues that I made throughout my university course and that helped me by spending time together, creating and discussing projects, and helping each other. For this reason, I thank Alberto, Mauro, Sabrina, and Chiara. Lastly, I want to thank my close relatives who supported me and tolerated my tech-enthusiast talks during these 3 years, and many more to come.

Abstract

This paper presents our exploration and comparison of deep learning architectures for predicting brain responses to realistic pictures. Inspired by the Algonauts Project's mission to foster collaboration between biological and machine intelligence researchers, we participated in the Algonauts 2023 Challenge focused on predicting brain responses to complex natural visual scenes. Leveraging the Natural Scenes Dataset (NSD) and collaboration with the Conference on Cognitive Computational Neuroscience (CCN), we had access to the largest suitable brain dataset.

Our research involved extensive experimentation with various models and datasets. Initially, we employed simpler models to predict brain activity but gradually introduced more complex architectures utilizing available data and embeddings generated by large-scale pre-trained models. We encountered challenges related to regularization and overfitting, especially with models based on specific embeddings. Despite these challenges, our models demonstrated a reasonable correlation with the available data.

Through iterative refinement, we found that employing multiple models, each dedicated to a region of interest (ROI) in each hemisphere of the brain of each subject, yielded the best results. These models shared a common architecture with a single fully connected linear layer. The input to this layer consisted of image embeddings generated by CLIP, and the output provided predictions for the activation levels of each vertex within the ROI.

While we surpassed the challenge baseline and achieved reasonable correlations, our results fell short of establishing a robust association with the data. This realization drives us to continue refining our approach, addressing challenges such as regularization and overfitting, and pushing the boundaries of our predictions. We remain eager to await the final results of the Algonauts 2023 Challenge while remaining confident that there is still ample room for improvement in predicting brain responses to realistic pictures.

Introduction

Algonauts

The Algonauts Project, which was first launched in 2019, has set forth on a mission to foster collaboration between biological and machine intelligence researchers. Their primary goal is to establish a shared platform where these brilliant minds can convene, exchange groundbreaking ideas, and embark on pioneering endeavors at the forefront of the intelligence frontier. Drawing inspiration from the remarkable explorations of astronauts venturing into the vastness of space, the "algonauts" have embraced a similar spirit of discovery as they delve into the realms of human and artificial intelligence, armed with state-of-the-art algorithmic tools. In doing so, they strive to propel both fields forward.

The Algonauts 2023 Challenge is specifically focused on the prediction of responses within the intricate human brain as participants perceive complex natural visual scenes. Through collaboration with the Natural Scenes Dataset (NSD) team, the Challenge was granted access to the largest suitable brain dataset available, thereby opening up unprecedented opportunities for the development of data-hungry models. The Algonauts 2023 Challenge has also forged a strategic partnership with the Conference on Cognitive Computational Neuroscience (CCN) to ensure its success and impact.

Computer Vision

In the early years of computer vision, researchers primarily relied on handcrafted features and traditional machine learning algorithms. These approaches involved manually designing and extracting features, such as edges, corners, or textures, from images. The extracted features were then used in various algorithms, e.g. decision trees, for tasks like object recognition or image classification. However, these methods had limited performance due to the difficulty of capturing complex visual patterns. Furthermore, they required domain-specific effort and were hard to generalize.

The introduction of Convolutional Neural Networks (CNNs) marked a significant breakthrough in computer vision. CNNs are deep learning models designed to automatically learn hierarchical representations from raw image data. They employ convolutional layers to capture local patterns and spatial relationships in an image. The advent of CNNs, such as AlexNet, demonstrated their ability to achieve state-of-the-art performance on tasks like image classification and object recognition.

As time went on, models got more complex and required a large amount of resources to train, making it difficult to train such models from scratch. Transfer learning emerged as an important technique to address these limitations. It involves leveraging pretrained models that have been trained on large-scale datasets, such as ImageNet. These models capture a wide range of visual patterns and can be fine-tuned or used as feature extractors for specific tasks. This approach allows even small datasets to benefit from the generalization power of models pretrained on massive datasets. Moreover, it makes it possible to achieve impressive results even when few resources are available.

A relatively recent and noteworthy entry was CLIP (Contrastive Language-Image Pretraining), an influential model developed by OpenAI in 2021. It combines vision and language understanding by training a single model on a large dataset of image-text pairs. CLIP can perform various tasks without task-specific fine-tuning, such as image classification, object detection, and image-to-text retrieval. It has shown impressive performance on benchmarks and enables powerful interactions between vision and language. The most relevant aspect of CLIP for this project has been its capacity to produce semantically rich embeddings from images.

Past approaches and notable works

Due to this year's challenge being not the first one from the Algonauts Project, we found it interesting to check out previous years' works.

The main goal of the 2019 Algonauts challenge was to predict brain activity from two sources, Functional Magnetic Resonance Imaging (fMRI) data from two brain regions or Magnetoencephalography (MEG) data from two temporal windows, using computational models. The brain activity was in response to viewing sets of images; for each image set, fMRI and MEG data were collected from the same 15 human subjects.
Unfortunately, the starting dataset, the aim, and the evaluation metrics were too different from what was required this year, making it difficult to get something useful out of the 2019 approaches for the 2023 challenge.

In 2021 the Algonauts challenge consisted in predicting brain responses recorded while participants viewed short video clips of everyday events. fMRI data was collected from 10 human subjects that watched over 1,000 short video clips. There were 2 challenge tracks: the Mini Track and the Full Track. The Mini Track focused on specified regions of the visual brain known to play a key role in visual processing. The Full Track considered responses

across the whole brain. Participants could play in the Mini Track, Full Track, or both. The task was: given the set of videos of everyday events and the corresponding brain responses recorded while human participants viewed those videos, train models to predict brain responses for videos for which brain data was held out.
This time around the challenge, specifically the Mini Track, was more similar to current year's one as they both had a focus on specific Regions Of Interest (ROIs) and the brain activity data included only fMRI data. However it's important to note that, due to the fact that the 2021 challenge was based on short video clips while the 2023 one was based on still images, there are some major differences that couldn't let us apply previous works as is. A key point common in all top three scoring submissions was taking into account different video features extracted from the video clips, with the best submission having an ensemble of models each tuned on a specific video feature.
This focusing on different features is something we also tried to implement, as will be explained in more detail later, by taking advantage of pretrained models to extract semantically rich features from images.

Apart from previous years' challenges, we also searched for other interesting approaches that could benefit our project and we found something exceptionally interesting: a study about high-resolution image reconstruction with latent diffusion models from human brain activity.
In this study the authors created and applied with good results models based on latent diffusion to reconstruct what a subject was seeing just by the fMRI brain activity. Other than being the opposite request of the challenge, as it goes from fMRI to image rather than from image to fMRI, it also uses the same dataset, NSD, on which the challenge is based.

Motivation and aim

Considering all the recent and novel studies, breakthroughs, and steps forward in this field to gain insight on how the human brain works by creating models that can effectively predict its activation based on certain specific inputs, and given the current wave of enthusiasm about artificial intelligence bringing further innovation and thus also improved computer vision models that can better create semantically rich embeddings suitable for a large variety of tasks, we decided to embark on this challenge provided by the Algonauts Project to compare different deep learning architectures to predict how the human brain behaves in response to seeing realistic and everyday pictures.

Available data

NSD dataset

The organizers of the Algonauts 2023 Challenge provided, in collaboration with the NSD team, the NSD dataset consisting of whole-brain, high-resolution (1.8mm isotropic, 1.6s sampling rate) fMRI measurements in ultra-high-field (7T) strength of 8 healthy adult subjects while they viewed thousands of natural scenes over the course of 30 to 40 scan sessions. For background, fMRI is a technique that measures brain activity indirectly by detecting changes in blood flow.

During the experiment, each subject viewed 10,000 distinct images, and a special set of 1000 images was shared across all subjects. This means the dataset contains data for 8 subjects times 9000 unique images plus 1000 shared images for a total of 73,000 images. Each of the 10,000 images was presented three times, for a total of 30,000 image trials per subject. Subjects were instructed to focus on a fixation cross at the center of the screen and perform a continuous recognition task in which they reported whether the current image had already been presented.

All images were presented with a visual angle of 8.4° ✕ 8.4°.

The fMRI data of the last three sessions of every subject is withheld and constitutes the basis for the test split that will be used to evaluate each submission.

The fMRI data is divided into the left (LH) and right (RH) hemispheres, each consisting of 19,004 and 20,544 vertices, with the exception of subjects 6 (18,978 LH and 20,220 RH vertices) and 8 (18,981 LH and 20,530 RH vertices) due to missing data.

The activity of each voxel was independently z-scored for each NSD scan session, and then the fMRI responses were averaged across repeats of the same stimulus images.

The fMRI data was preprocessed, in particular to remove head motion and to upscale from 1.8mm to 1mm.

An important note is that the NSD dataset images are cropped versions of COCO dataset pictures.

For more details on the dataset, it's possible to refer to the original NSD website.

This dataset was the base for the challenge and the development and application of deep learning models, as it contains the images to train on, the corresponding brain activity to train and evaluate the models we created, and the images for which the brain activity data was withheld to prepare the models for submission.

COCO dataset

The COCO (Common Objects in Context) dataset is a widely used benchmark dataset for object detection, segmentation, and captioning tasks in computer vision research. It was created to provide a standardized and comprehensive dataset for advancing the state-of-the-art in these areas.

The COCO dataset consists of a large collection of images, each of which is annotated with various types of information. The dataset contains images that capture a wide range of objects in different contexts, including everyday scenes, people, animals, vehicles, and more. The annotations in COCO provide detailed information about the objects present in the images, such as bounding box coordinates, segmentation masks, and labels.

The dataset is divided into three main subsets: training, validation, and test. The training set is the largest and is typically used to train object detection or segmentation models. The validation set is used for model selection and hyperparameter tuning during the development process. The test set is kept private and is used for evaluating the performance of models in a standardized manner.

In particular, each image is annotated with several captions, which could prove useful when predicting brain activity, as when a human being sees a picture, many other pieces of information come to mind, and a text description of what is being seen could represent an idea that forms in the subject when seeing a certain image.

AlexNet

AlexNet is a CNN architecture that has played a significant role in the advancement of deep learning and computer vision. It was trained on the ImageNet dataset, consisting of millions of labeled images across thousands of categories, and achieved a significant reduction in error rates compared to previous models.

PyTorch provides a pretrained version of AlexNet, and it is used in the base development kit provided by the Algonauts 2023 Challenge organizers.

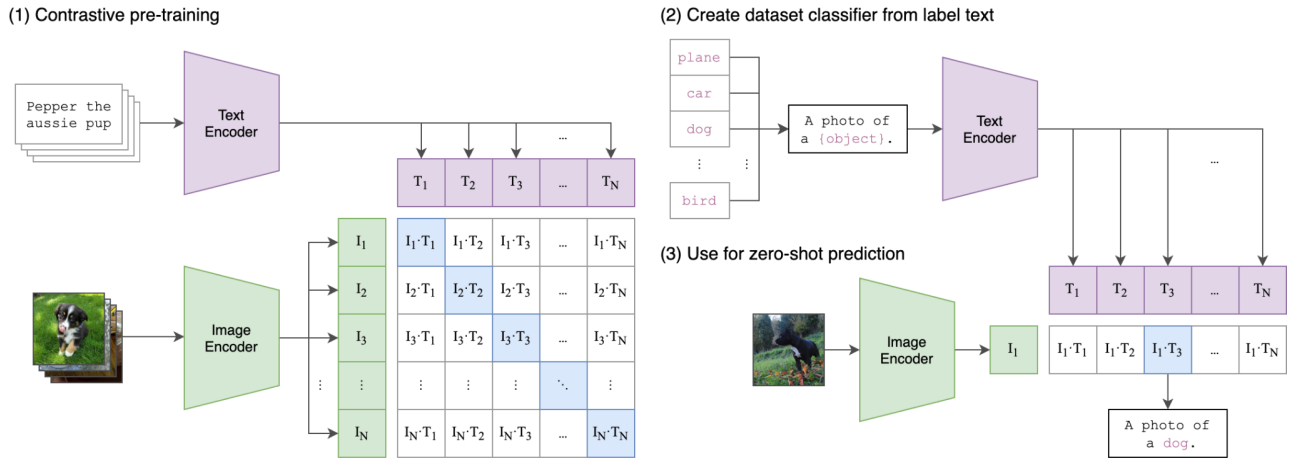
The reason behind this choice is that it’s possible to extract the output values of any layer of the pretrained AlexNet when processing an image, which will be rich in information about the image itself and its contents. Hopefully, these features will resemble how the brain processes image information and will help another deep learning model take them as input and output the corresponding brain activity. It goes without saying that this is mostly a toy example due to its simplicity, and the yielded results are not the best.

CLIP

As anticipated before, CLIP is a model developed by OpenAI in 2021.

The key idea behind CLIP is to train a single model to understand both images and text in a unified manner. This allows CLIP to perform a variety of vision and language tasks without task-specific fine-tuning.

During pretraining, CLIP learns to associate images and their corresponding captions by maximizing the similarity between positive pairs (an image and its caption) while minimizing the similarity between negative pairs (an image and a randomly sampled caption). This contrastive learning approach encourages the model to capture the semantic relationship between images and text.



One of the main strengths of CLIP is its ability to generalize across different tasks without requiring task-specific fine-tuning. It can perform tasks such as image classification, object detection, and zero-shot image-to-text retrieval, where given a textual prompt, it can retrieve relevant images from a large dataset without any specific training for that particular prompt. CLIP has demonstrated impressive performance on various benchmarks and has shown the potential for cross-modal understanding. By learning a joint representation of images and text, CLIP can generate semantically rich embeddings. These embeddings are what we were after, to apply transfer learning based on CLIP too. Being CLIP a more powerful and recent model than AlexNet, the hope is that the produced embeddings will exhibit enhanced semantic understanding and capture richer visual representations. Furthermore, the textual embeddings that CLIP could produce with the COCO dataset image captions as input could result in adding more semantic meaning to what the image embeddings represent.

The only issue with textual embeddings is that they are of variable length, while the image embeddings result easier to use because they have a fixed length of 512.

SAM

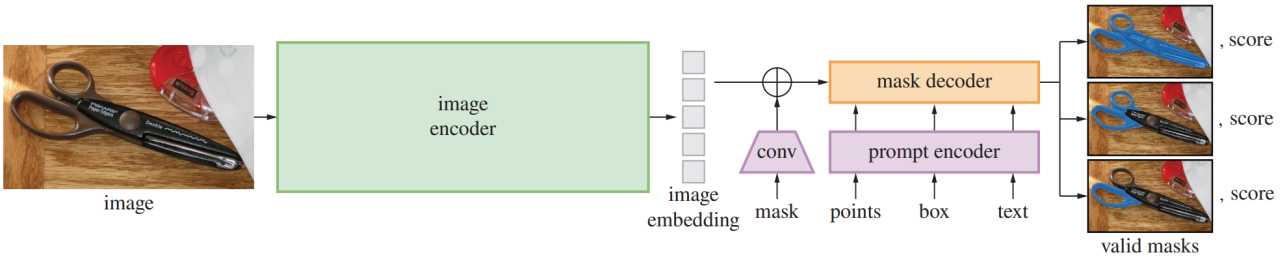
SAM (Segment Anything Meta) is a model developed earlier this year by Meta to create valid segmentation masks given any prompt for any image. This task is, however, inherently ambiguous.

Simply put, generating a semantically valid mask means that even when a prompt is ambiguous and could refer to multiple objects, the output should be a reasonable mask for at least one of them.

The concept behind it comes by trying to translate the idea of a prompt from Natural Language Processing to segmentation, expecting then, like a language model, to output a coherent response to an ambiguous prompt.

It was trained on 11M images and 1B masks, reaching quite good results even on zero-shot predictions. It also managed to do very well on edge prediction, a task that it was not trained for.

A simplified version of the SAM architecture is illustrated below.



The image encoder mentioned here refers to a powerful computational model known as a pre-trained Masked AutoEncoder Visual Transformer (MAE ViT). This specific MAE ViT has been fine-tuned specifically to handle high-resolution input images, making it capable of processing detailed visual information effectively. The MAE ViT runs only once for each image, which allows for efficient utilization when preparing the model for inference.

The outputs generated by the image encoder are image embeddings. These embeddings encapsulate the essential characteristics and semantic information of the input image. With a width and height set to 64, and 256 channels, these embeddings provide a compact representation of the visual content of the image.

In a similar fashion to CLIP, the image embeddings generated by the image encoder possess semantic richness. By utilizing these embeddings as a starting point, the subsequent stages of processing can benefit from a more nuanced understanding of the image content compared to utilizing features extracted by previous models like AlexNet.

Work methodology

Tools

Five essential tools were used for this project, along with a development kit provided by the Algonauts organizers.

1. Secure Shell (SSH)

   Secure Shell, commonly referred to as SSH, is a cryptographic network protocol that provides secure remote access to systems over untrusted networks. Since the machine we worked on was physically located at the university and was shared between different projects, SSH allowed us to log in remotely and securely.
   It enabled us to work remotely, a capability particularly valuable for this project, and it also allowed us to easily share the remote machine with other teams that needed to work on other projects.

2. tmux

   tmux, short for terminal multiplexer, is a command-line tool that enhances productivity by allowing users to create and manage multiple terminal sessions within a single window. The benefits of tmux are manifold. First, it enabled us to run multiple commands or programs simultaneously within a single terminal window, eliminating the need for multiple terminal tabs or windows. This capability enhances multitasking and facilitates efficient session management, as we could switch between different tasks or projects effortlessly. Additionally, tmux supports session persistence, allowing developers to detach from a session and reattach to it later, even after closing the terminal or disconnecting from the SSH session, which is extremely useful for running scripts overnight or across multiple days.

3. Visual Studio Code

   Visual Studio Code is a powerful and extensible source code editor developed by Microsoft. Its popularity among developers stems from its rich features, ease of use, and extensive plugin ecosystem. VSCode provides a highly customizable and integrated development environment, offering features such as code completion, syntax highlighting, debugging, and version control integration.
   One of the notable strengths of VSCode is its remote development capabilities. With the help of extensions like "Remote - SSH," developers can leverage SSH to work on remote machines directly from within the VSCode interface. This feature

enables developers to write, test, and debug code on remote servers seamlessly, combining the convenience of a local development environment with the power of remote resources.

4. Git and GitHub

   Git is a distributed version control system, enabling developers to track changes to their codebase, collaborate with others, and manage different versions of their projects. One popular platform built on top of Git is GitHub, which provides a web-based interface for hosting Git repositories and facilitating workflows.
   Git enables efficient and seamless collaboration by allowing multiple developers to work on the same codebase simultaneously.
   GitHub, as a web-based hosting service, provides additional functionalities and benefits to developers. It offers a central repository for storing Git repositories, making it easy to share code with others, along with many other features.
   By utilizing Git and GitHub in conjunction with SSH and VSCode, we could establish a robust and efficient development environment. We could collaborate effectively by sharing the codebase, tracking and managing code changes, and reviewing and discussing code improvements.

5. Conda

   Conda is a popular package and environment management system used primarily in the Python ecosystem. It allows developers to create isolated environments containing specific versions of Python along with the necessary dependencies and libraries. These environments enable reproducibility and eliminate conflicts between different projects' dependencies. Conda simplifies the process of installing, managing, and updating packages and libraries. With it, it’s possible to create, activate, and deactivate environments, ensuring that each project has its own isolated set of dependencies.

6. Development Kit

   The Algonauts 2023 Challenge organizers provided a development kit tutorial of the Challenge where they showed how to load and visualize the Challenge fMRI data and the stimulus images, build linearizing encoding models using a pretrained AlexNet architecture, evaluate them and visualize the resulting prediction accuracy, and finally prepare the predicted brain responses to the test images in the right format for submission to the Challenge leaderboard.

Software

Given its ease of use and the rich ecosystem around it, we chose Python as the programming language for this project. Python is known for its simplicity, readability, and versatility, making it an excellent choice for both experienced developers and beginners. Its extensive library support and vast community contribute to its popularity, enabling developers to find solutions to various challenges quickly.

Furthermore, to incentivize this choice even more, the Development Kit already provided a starting point in Python. This made it convenient to build upon existing code and leverage the features and functionalities offered by the Development Kit seamlessly. By utilizing Python from the beginning, we were able to reduce development time and focus more on the core objectives of our project.

Along the same lines, PyTorch, a popular deep learning framework, was used in the Development Kit. PyTorch offers a user-friendly interface and provides easy access to creating, training, and evaluating machine learning models. It supports dynamic computation graphs, allowing for efficient model prototyping and experimentation. With PyTorch, we could rapidly iterate on different architectures and hyperparameters, accelerating the development process.

Moreover, PyTorch's integration with NVidia GPUs was a significant advantage for our project. NVidia GPUs are renowned for their parallel processing capabilities, making them ideal for accelerating deep learning computations. PyTorch seamlessly utilizes these GPUs, enabling us to leverage the power of the ones we had and significantly reduce the time required to develop and test our ideas.

Other Python libraries were used, to name the major ones:

- Matplotlib, to create and plot metrics graphs easily;
- NumPy, to operate on large amounts of data and easily interface with some of the Challenge data;
- Nilearn, a machine learning module for neuroimaging, handling some of the Challenge data and allowing to plot values on a 3D brain map;
- Clip and Segment Anything for their respective models,
- Jupyter Notebook, in the first phases to visualize plots directly along with the code and while running it.

Hardware

The available hardware consisted of a remote machine with:

- Intel Core i9-13900K 24c (32t) up to 5.80GHz with 32MB L2 cache and 125 W TDP
- 128GB RAM
- 2x NVidia GPUs RTX 3090 Ti 10752 CUDA cores up to 1.86GHz, with 24GB VRAM, 384-bit bus, and 450W TDP

Approach

The Challenge is relatively new, marking merely its third iteration, and with substantial distinctions each time. Furthermore, the domain of neuroscience falls beyond the scope of our expertise and I must emphasize that this project was one of my initial forays into the realm of AI development.

In light of this, our initial step was to seek different ways to comprehend the provided data, delving into the dataset itself and examining its structure, and dedicating effort to acquiring an understanding of fMRI and its specificities.

Subsequently, our focus shifted towards exploring and identifying other approaches, venturing into related disciplines and past competitions in search of potential sources of inspiration to guide our decision-making process.

Following this research phase, we began developing starting from the Development Kit provided with the Challenge and experimenting with different ideas we had previously come up with. Each time we made progress, we made sure to accurately document its results for later use and comparison with new models, and every time we reached a milestone, we reorganized the codebase to improve generalization and efficiency.

While the hardware specifications may indeed appear impressive for consumer platforms or small-scale projects, it is imperative to consider the challenging circumstances arising from the machine's shared utilization among numerous users and projects. Given the substantial volume of data to be processed and the intricate nature of our models, meticulous planning was indispensable to ensure optimal efficiency and judicious allocation of memory resources.

As we were nearing the end of the time we set for this project, which almost coincided with the deadline to submit our predictions to the Challenge leaderboard, we began the final phase of testing and improving our most interesting model.

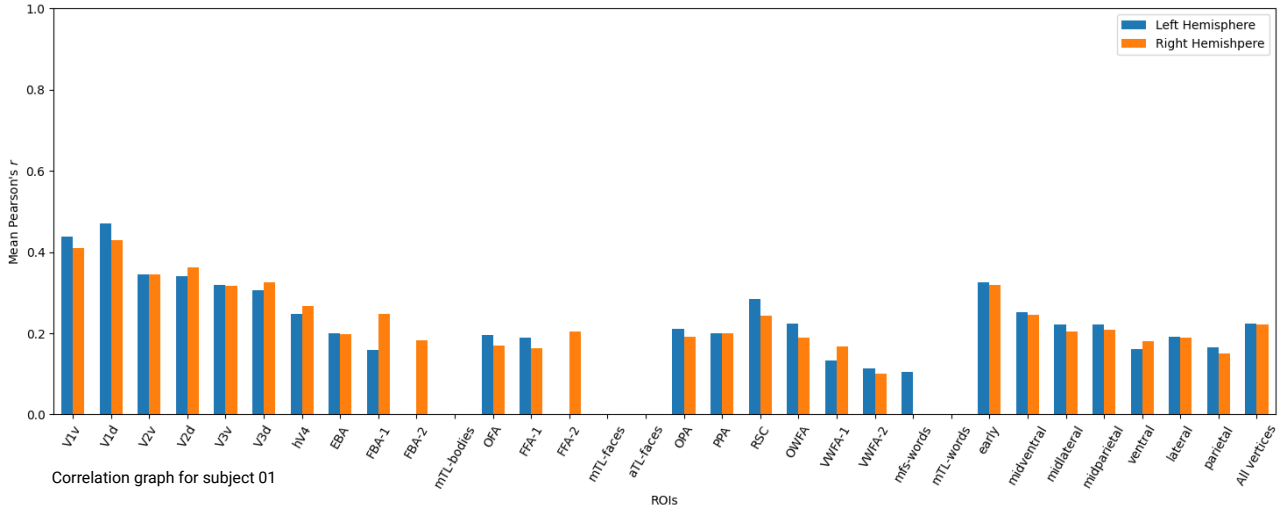
Developing course and experiments

Introduction

We performed many experiments, but not all are reported here, only the most relevant.
Across all experiments, the loss function used was the MSELoss provided by PyTorch.
Not in all, but in most experiments we kept track of the following metrics: correlation, training loss, validation loss, subjects' individual losses and correlations (when training for multiple subjects).

Challenge Baseline

The starting point of development was the Challenge Development Kit.
It provided basic data handling and a first model that served as a baseline to compare our next architectures, along with a way to evaluate the performance of models.
The baseline architecture was a simple linear regression fit on the image feature embeddings extracted from the second 2D convolutional layer of AlexNet.
The following graph illustrates the correlation metric value for the baseline model:



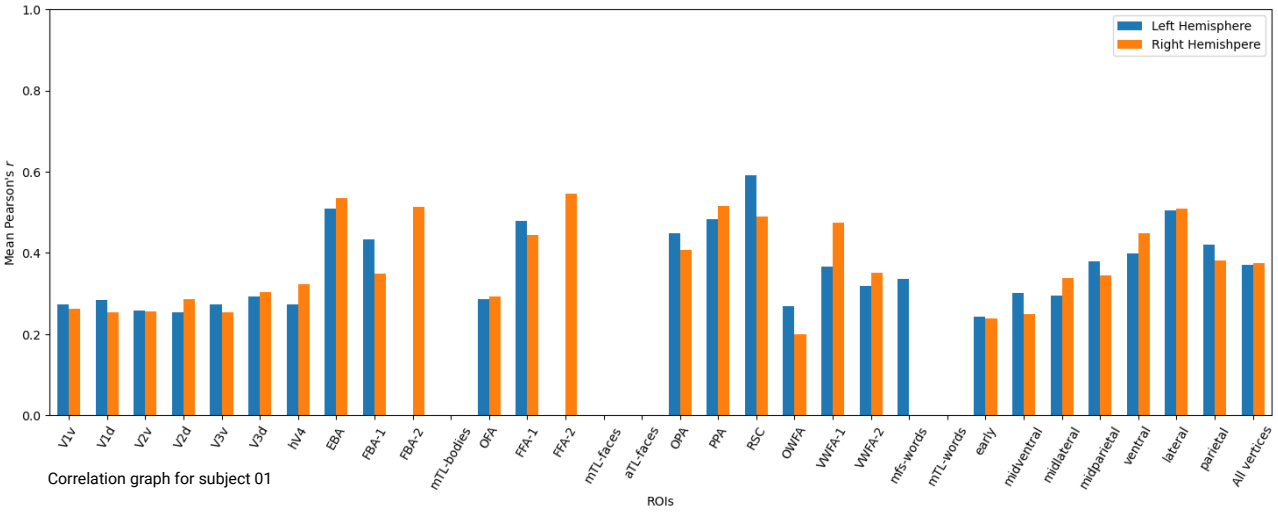
Correlation graph for subject 01

This architecture took as input image embeddings extracted from AlexNet and predicted with a linear regression the whole brain activity (from now on, "whole brain" refers to the set of all ROIs).
It's plain to see that it doesn't look great; not even a single ROI managed to get past a 0.5 correlation with the ground truth, and the average was even lower, at less than 0.3.
It's important to note that this graph refers to the training and predictions on a single subject, in particular subject 01. In addition, the features extracted from the 2D convolutional layer of AlexNet are flattened before being fed to the linear regression.

After dedicating a considerable amount of time to researching different computer vision models that could offer potentially better features, particularly among novel models. Out of the many models we looked at, two caught our attention: CLIP and SAM. These models showcased remarkable prowess in generating image features, yielding results that were exceptionally good in their scope. Intrigued by their promising potential, we resolved to deepen the use of CLIP and SAM as image features generators for our experiments.

CLIP-LinReg

Our first experiment, named CLIP-LinReg, consisted of taking the same baseline model, but instead of feeding to the linear regression features from AlexNet, we fed embeddings extracted from CLIP. CLIP has the advantage of creating embeddings very rich in semantics despite being only a single tensor with shape [512].
The results we got, for the same subject as before, were these:



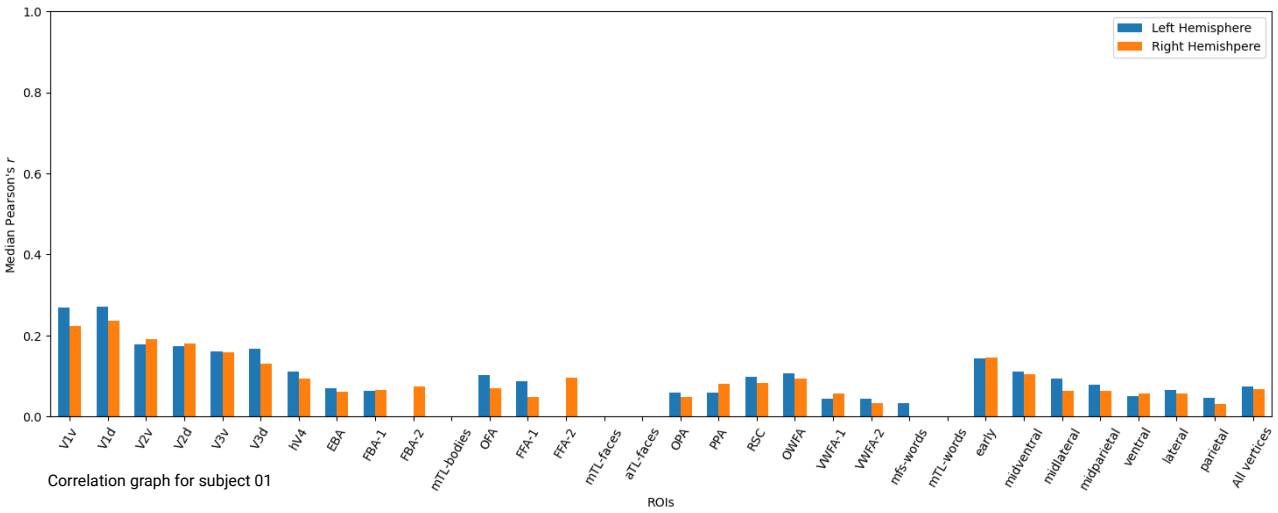
This was unexpectedly better than we thought it would be, with 5 ROIs above the 0.5 correlation threshold and an average correlation of about 0.4, more than a 0.1 increase over the linear regression based on AlexNet. It's important to note, however, that despite the average being higher, some ROIs faced a decrease in their correlation score.
This suggests that, while CLIP's embeddings are better overall, for some specific areas, AlexNet's ones appear to be more suitable, maybe due to the latter being more sensitive to the spatial and visual characteristics of the image whereas CLIP ones mainly represent semantics.

SAM-LinReg

This experiment was not actually performed, as SAM embeddings (from now on, “SAM encodings” refer to the embeddings from SAM fine-tuned ViT image encoder) are too large to be flattened and fed into a linear regression model.
SAM embeddings for 1 image consist of a tensor with shape [256, 64, 64].
For this reason, SAM embeddings needed some preprocessing, and we avoided experimenting with them further until we started applying more complex models that could handle that large amount of data, for instance by applying some 2D convolutional layers.

AlexNet-Simple

This experiment was a way to test out how introducing a more complex model would change the predictions, to see whether it would come with some improved performance.
The model architecture was a simple Linear layer from PyTorch, taking as input AlexNet features and giving out whole brain activity predictions.

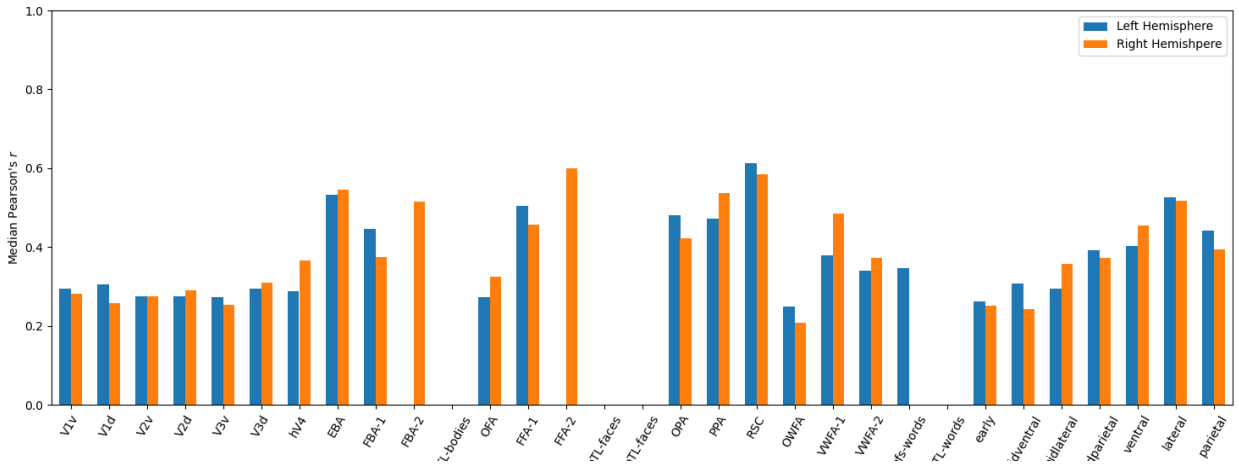


Needless to say, the graph shows a diminished performance. However, the bars are similarly “shaped” to the Challenge Baseline results, indicating that most probably the hyperparameters chosen for the experiment were not optimal.
However, to speed up our process, we decided to go on with the experiments.
Below the hyperparameters configuration table:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	1e-3	0.5	10 epochs	0.9	0.999	1e-8	1e-2

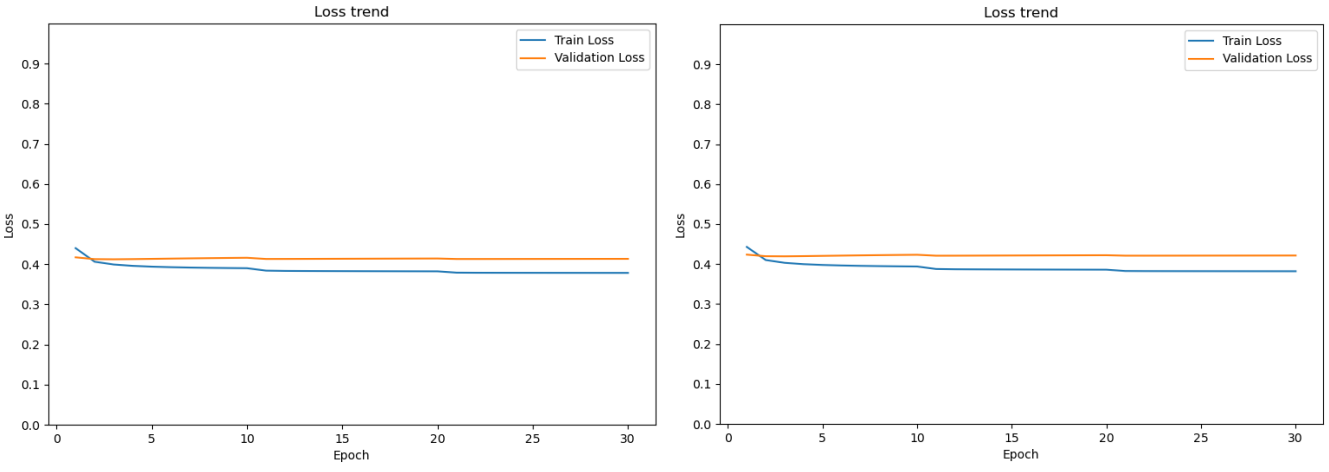
CLIP-Simple

Along the same line of the previous experiment, we tested out the same model architecture with the difference that it would take CLIP image embeddings as input. This time around, the results were almost identical to the CLIP-LinReg ones:.



Correlation graph for subject 01

These results suggest that, other than confirming CLIP image embeddings to be richer in semantics, they are also more robust and thus less sensible about hyperparameters. This model also managed to avoid overfitting for the most part, as shown by these graphs of train and validation loss for the left and right hemispheres:



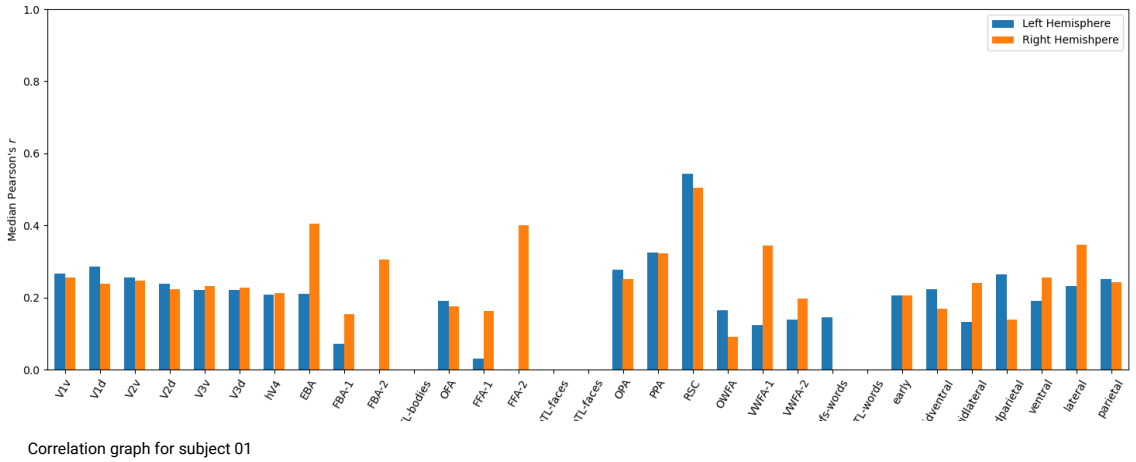
The graphs refer, from left to right, to the left and right hemispheres of subject 01

The hyperparameters were the same as the previous experiment:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	1e-3	0.5	10 epochs	0.9	0.999	1e-8	1e-2

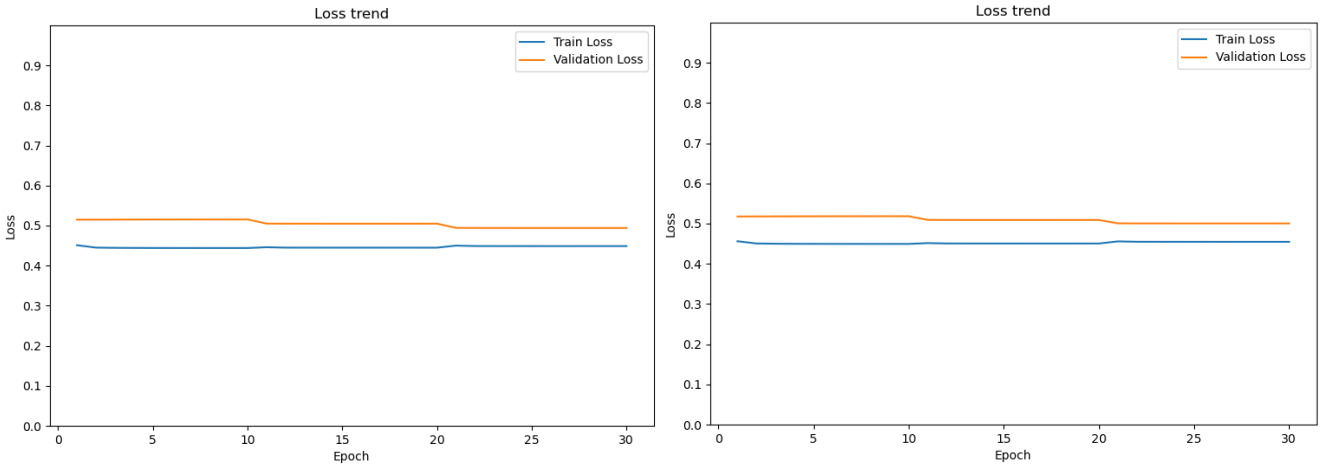
CLIP-Simple-All

This experiment involved the same architecture and configuration as CLIP-Simple, but trained with all available data of all subjects. These results indicate that it's hard to generalize across subjects, with some ROIs facing a fall in correlation more than others.



Correlation graph for subject 01

We used the same set of hyperparameters as before. Despite the fall in correlation, this time too the model didn't overfit as train and validation loss kept decreasing as epochs went on:



The graphs refer, from left to right, to the left and right hemispheres of subject 01

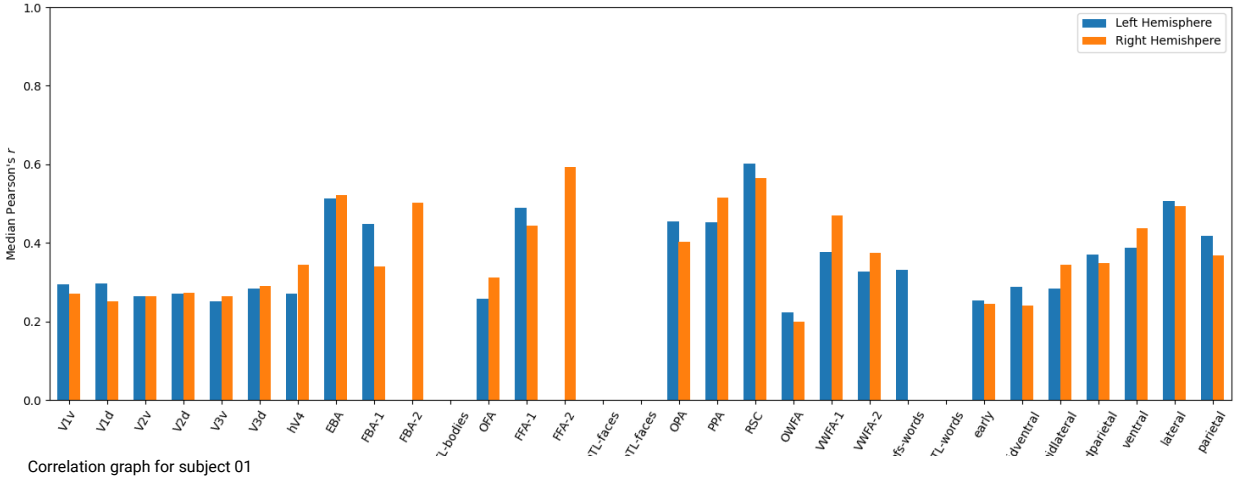
AlexNet-DNN

After exploring a single linear layer, we wanted to try to add more layers of the same kind and introduce some non-linearity between them.

However, as with the SAM-LinReg experiment, convolutional features extracted from AlexNet were too large to be processed by a fully connected Deep Neural Network, and for this reason, this experiment was also not performed.

CLIP-DNN

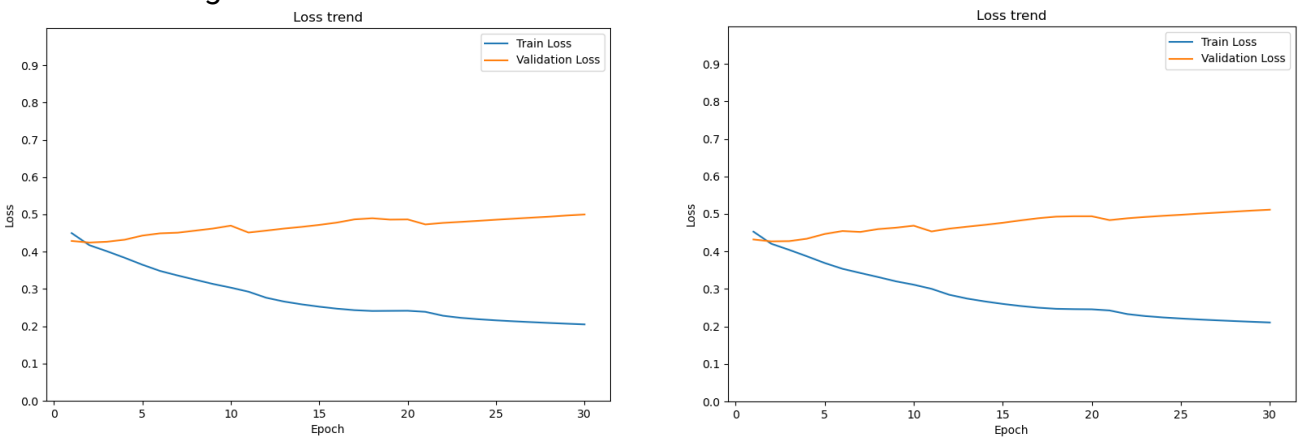
As CLIP is the only selected image features generator that could produce embeddings not too large, we proceeded to create a new fully connected Deep Neural Network architecture for it, namely 4 linear blocks and one linear layer, where a linear block consisted of a sequence of 3 layers: Linear, BatchNorm1d and LeakyReLU..



Correlation graph for subject 01

Although it's not easy to see using the graphs only, the numerical data that we produced along them proved that this architecture produced on average slightly poorer results, while also taking more time to train and evaluate.

Furthermore, from the graphs it's easy to see how this model, given it's higher complexity, was overfitting:



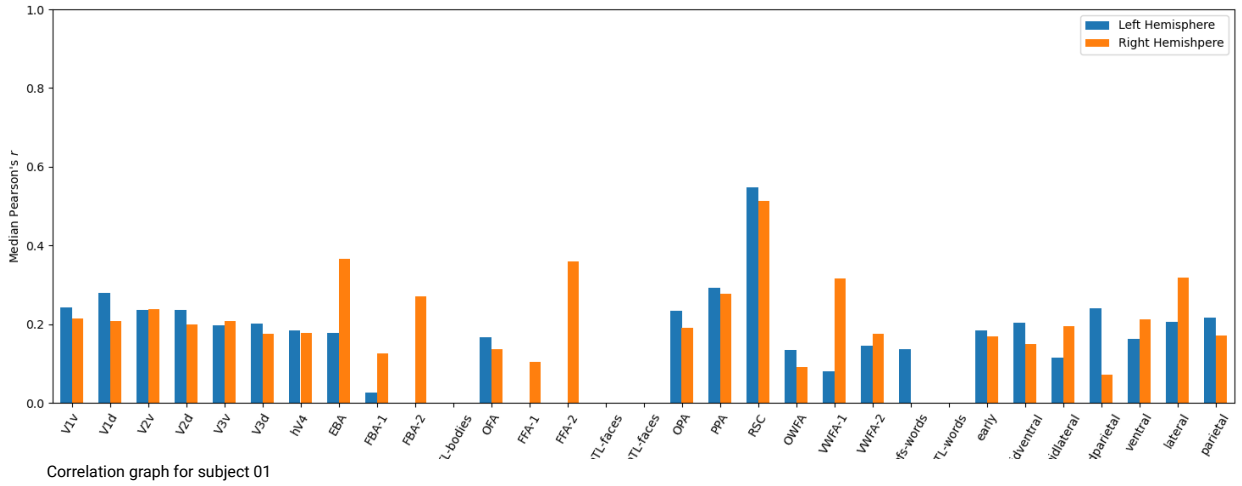
The graphs refer, from left to right, to the left and right hemispheres of subject 01

We also found out, and the results above refer to this configuration, that the best set of hyperparameters for this architecture was:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	1e-3	0.5	10 epochs	0.9	0.999	1e-8	1e-2

CLIP-DNN-ALL

With the same architecture as in the previous experiment, CLIP-DNN, we trained the model on data of all subject, and this time too we found out that generalizing across multiple subjects is hard:



Correlation graph for subject 01

Positional Encoding

This experiment was a test for a change of approach we wanted to try. The idea behind it was to predict each ROI separately, one vertex at a time. Each time, the model, along with the image embeddings, would be fed a positional encoding of the vertex position within the current ROI.

Before jumping into this, however, we wanted to make sure the model could actually benefit from the new positional information. This is what this experiment was for: to search for the best way to create positional encodings and to verify that it was possible to reconstruct the encoded positional information from them.

The positional information we wanted to encode was the following: x, y, and z coordinates of the vertex with respect to the center of the ROI's bounding box, along with the distance from the center itself.

We based our positional encoding on the one described in section 3.5 of the paper *Attention Is All You Need*. Each coordinate is encoded with a sinusoidal encoding:

$PE_i = sin(\frac{pos}{100^{\frac{2i}{dim}}}); PE_{2i} = cos(\frac{pos}{100^{\frac{2i}{dim}}})$ where the resulting array is the encoding for a single coordinate, PE_i is an element of such encoding, pos is the actual coordinate value, and dim is an hyperparameter that indicates the dimension (length) of the encoding.

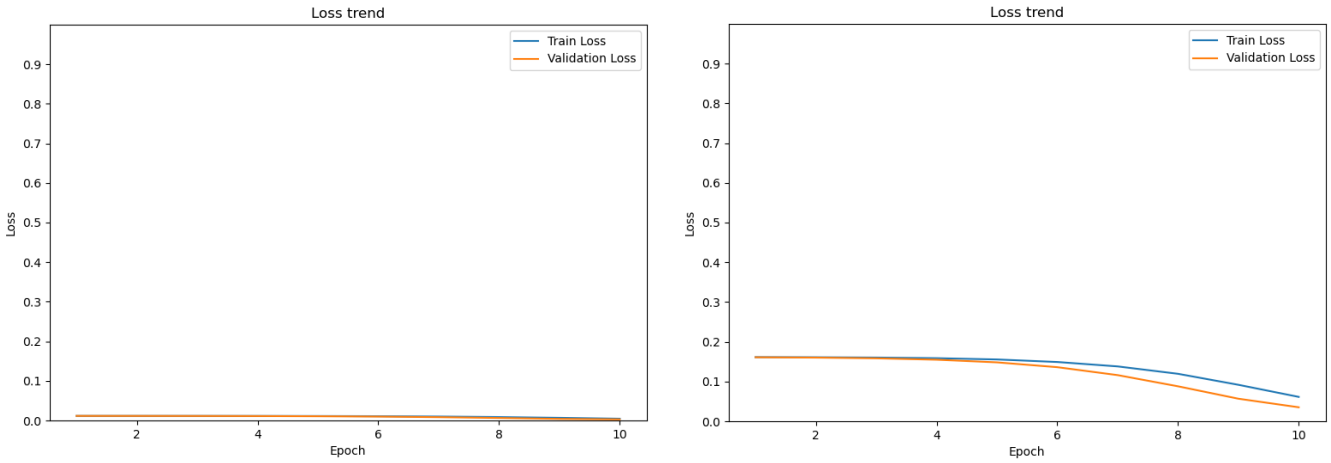
The pos value, the coordinate, is obtained by creating a bounding box of the ROI, shifting the reference system such that the origin is the center of the ROI's bounding box, expanding by a factor of 32 the space of the bounding box (since vertex are not uniformly distributed in the ROI's bounding box, expanding ensures that there is a meaningful difference in coordinates between each vertex), and getting either the x, y, z, or the distance from the origin.

What we searched for was the best value for the dimension of the positional encoding and the number of linear layers in the model architecture.

To do this, we ranged from 4 to 8 fully connected linear layers with step size 2, with the model structured similarly to the CLIP-DNN experiment, and the dimension of the positional encoding ranging from 16 to 64 with step size 8.

We found out that the configuration that could best reconstruct positional information was the one with 6 fully connected linear layers and a positional encoding dimension of 48.

The following are two examples of train and validation loss for two different ROIs, one with around 3,000 vertices ("early", right) and one with around 400 vertices ("OFA", left), of subject 01 of the left hemisphere, but the situation is similar for other ROIs, other subjects and both hemispheres:



Bigger ROIs generally had better results than small ones, indicating that with small ROIs there is less spatial difference across vertices and thus less differentiation between them. Expanding the bounding box when encoding the position was key for this reason.

It was a common pattern to have, near the end of the training epochs, a faster decrease in validation loss than train loss, which indicates that the model is correctly generalizing and is not overfitting at all.

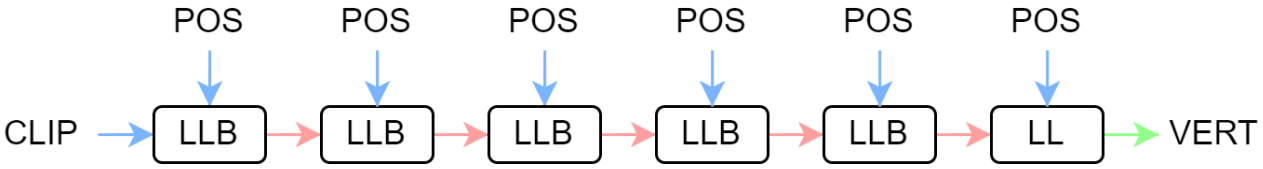
These results strongly suggest that the chosen way to encode positional information is valid and the model has enough power to represent and extrapolate all the information.

CLIP-Pos

Given the results from the previous experiment, we tried to combine the encodings from CLIP with the positional encodings.

The architecture chosen was the one that yielded the best results with reconstructing positional information: a series of 5 linear layer blocks (LLB) and 1 final linear layer (LL), where a linear layer block is given by a Linear layer, a BatchNorm1d, and a LeakyReLU.

CLIP encodings were processed at each step by the model, and each time they were concatenated to the original positional encoding.

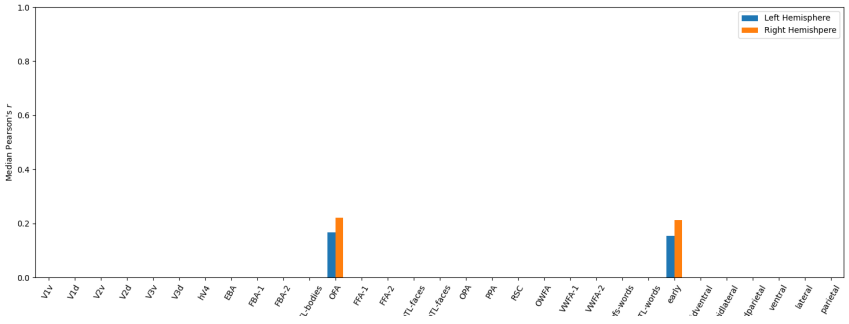


After predicting all vertices of a ROI for a given image, the output vertex activations were concatenated and compared with MSE loss to the actual ROI activation for that image.

We trained the model with nested batches of all ROI vertices for 16 images, and we trained it with data from all subjects.

Since this model was big and would take quite some time to train, we trained and evaluated it only on 2 ROIs of subject 01, one among the biggest ones (“early”) and another among the smallest ones (“OFA”).

Unfortunately, despite our best efforts to fine-tune the hyperparameters, this model didn’t give the results we hoped for, performing considerably worse than the best and simplest one we had so far.



For ease of comparison, our best model so far had correlation values for the left hemisphere of 0.27 for OFA and 0.26 for early, while this model had seen more than a 0.1 worsening with 0.167 for OFA and 0.155 for early.

The hyperparameters we found were best for this architecture were these ones:

Optimizer	Learning Rate	β_1	β_2	ε	Weight decay
AdamW	1e-3	0.9	0.999	1e-8	1e-2

SAM-Pos

Despite the previous experiment not yielding the results we had hoped for, it was only one of the first attempts to leverage the positional encodings.

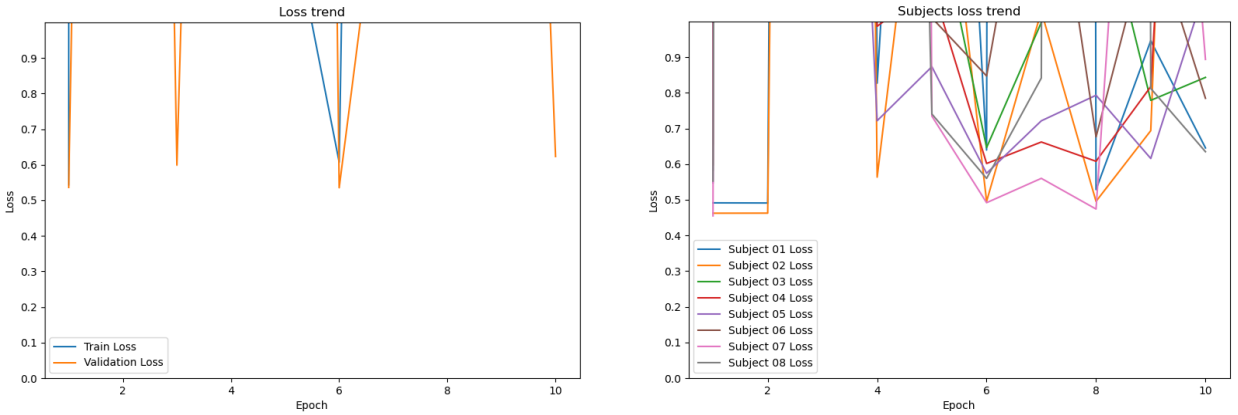
This time we tried to use SAM-generated embeddings; however, we had to change the model architecture to account for the difference in shape of the SAM embeddings tensor compared to CLIP ones'.

The change was to add, before the linear layers, some 2D convolution to further process SAM embeddings and have a shape that was feasible to feed into fully connected layers. In particular, we added 3 2D convolutional layers (kernel size 3, stride 1) with, in between each, a sequence of BatchNorm2d, LeakyReLU, and MaxPool2D (size 2, stride 2). This reduced the size of the embeddings tensor from [256, 64, 64] to [256, 6, 6]. This tensor was then flattened and fed into the rest of the model, along with the positional encodings that were not present before. Another advantage of the 2D convolutions part was that it needed to run only once per image, while the rest of the model had to run once per ROI vertex. By carefully handling the output of the convolutions, we could save a lot of time in training.

This time too, the model was trained on all available data of all subjects, but only on the early and OFA ROIs to save time. We used the same set of hyperparameters:

Optimizer	Learning Rate	β_1	β_2	ε	Weight decay
AdamW	1e-3	0.9	0.999	1e-8	1e-2

We found out that the model in this configuration was highly unstable, which was evident by the graphs of train and validation loss:



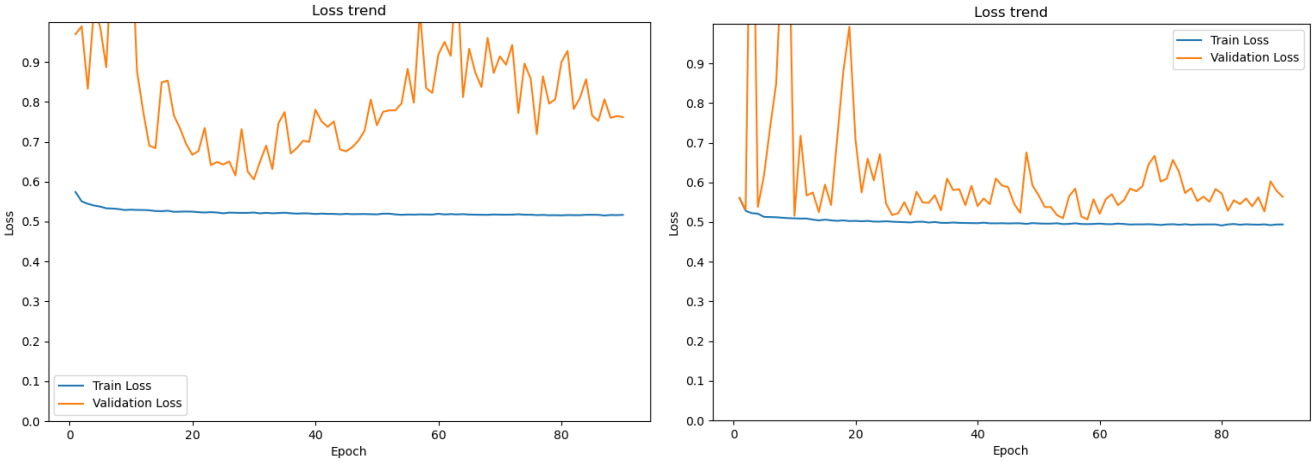
The graph to the left shows the trend of train and validation losses averaged across multiple subjects for OFA ROI, while the graph to the right shows the train loss for each subject.

SAM-Conv1d

To speed up further the training in our previous experiment, we decided to change the linear layers into 1D convolutional layers with both kernel and stride 1, which should have almost the same effect while being easier to parallelize. Indeed we saw a speed up of about 3 times while training.

We also added some regularization with a 20% dropout layer in between each 2D convolution layer, after the max pooling. Furthermore, we increased by a factor of 1.5 at each step the number of channels in the 2D convolution, to allow the model to capture many more different patterns. Finally, we drastically decreased the learning rate and added a decrease of it while training, along with 9 times more epochs than before.

As we had strong enough evidence that generalizing across multiple subjects was too hard, we trained only on subject 01. This time the results were definitely better:



From left to right, the graphs for OFA ROI of train and validation loss for respectively the left and right hemispheres of subject 01. Although it's surely better than before, especially the train loss, it's evident that this model still struggles to generalize and it's reflected in the validation loss that oscillates way too much.

The following were the hyperparameters used for the best configuration we could find:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	5e-6	0.9	3 epochs	0.9	0.999	1e-8	1e-2

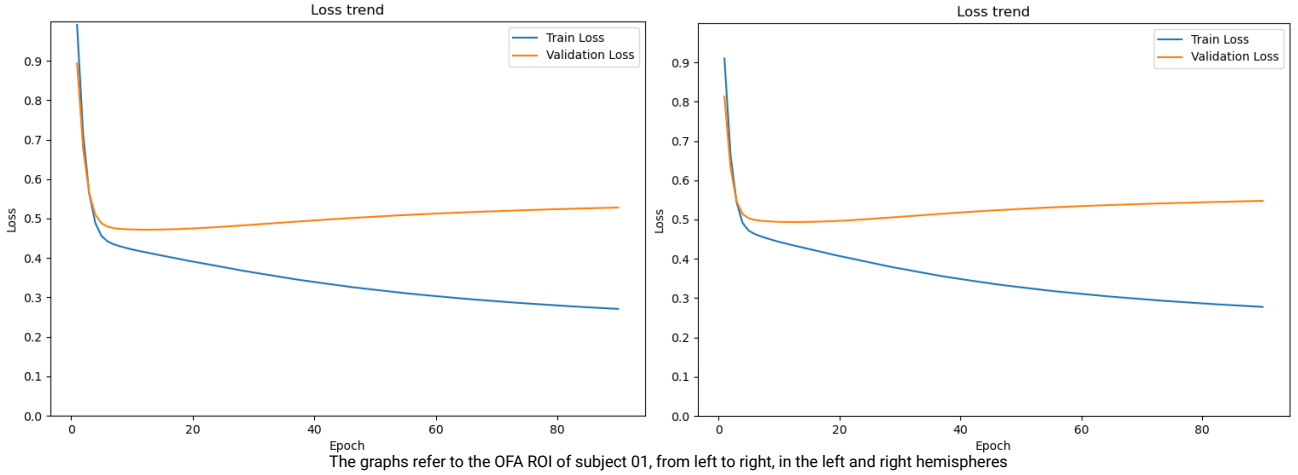
With these experiment results, we discarded SAM encodings as all models we tried (which may not seem many, but as mentioned before, we reported here only the most relevant ones) were not stable, and the ones that were would have very high loss values.

CLIP-Conv1d

Since introducing 1D convolutions in place of fully connected linear layers sped up the training time, we could experiment with more layers and more hyperparameters sets.
We replaced CLIP-Pos linear layers with 1D convolutional layers with both kernel size and stride equal to 1. Our first attempt was with the following hyperparameters:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	5e-6	0.9	9 epochs	0.9	0.999	1e-8	1e-2

But the model was still highly overfitting:

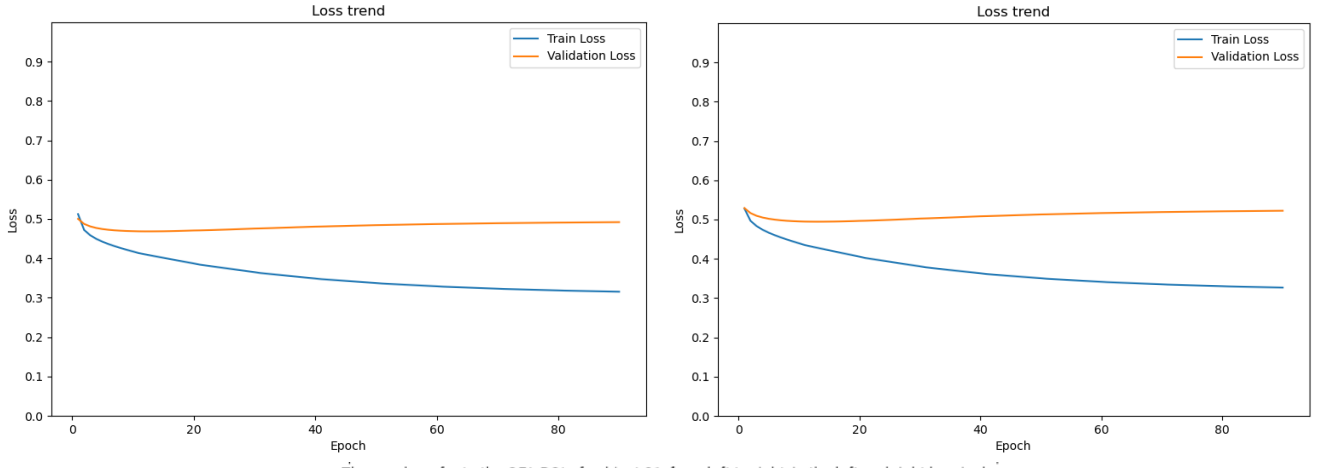


The graphs refer to the OFA ROI of subject 01, from left to right, in the left and right hemispheres

Our best attempt with this same architecture was with the following hyperparameters:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	5e-6	0.75	10 epochs	0.9	0.999	1e-8	1e-2

This time the model overfitted slightly less and managed to get slightly better correlations:



The graphs refer to the OFA ROI of subject 01, from left to right, in the left and right hemispheres

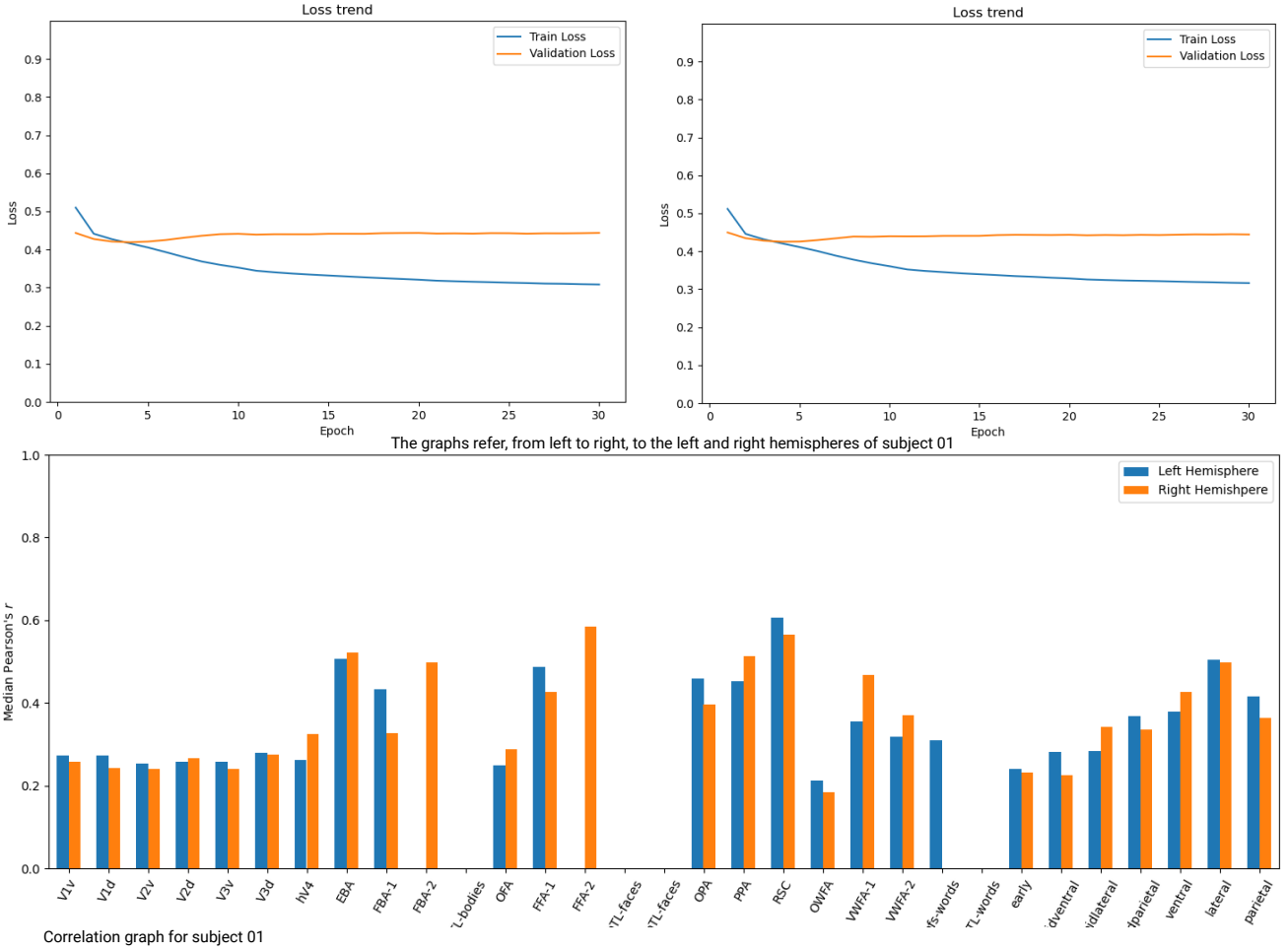
CLIP-Dropout

To prevent the overfitting seen in the previous experiments, in particular CLIP-DDN and CLIP-Conv1d, we added some more regularization by introducing dropout between some of the linear/1D convolutional layers.

Starting with the dropout version of CLIP-DNN, our best attempt introduced a 5% dropout layer between each linear layer and with this hyperparameters set:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	1e-4	0.5	10 epochs	0.9	0.999	1e-8	1e-2

This configuration had the following results:



Despite overfitting less than the original CLIP-DNN, the validation loss was a bit higher and thus the correlation scores were a bit lower. The regularization, instead of closing the distance between train and validation loss by shifting the latter, seems to have had the effect of increasing the train loss. Unfortunately, although we explored different values for the dropout and the hyperparameters, we couldn't get it to surpass the original model.

To improve the model architecture used in the CLIP-Conv1d experiment, we introduced a less frequent, but more aggressive, dropout.

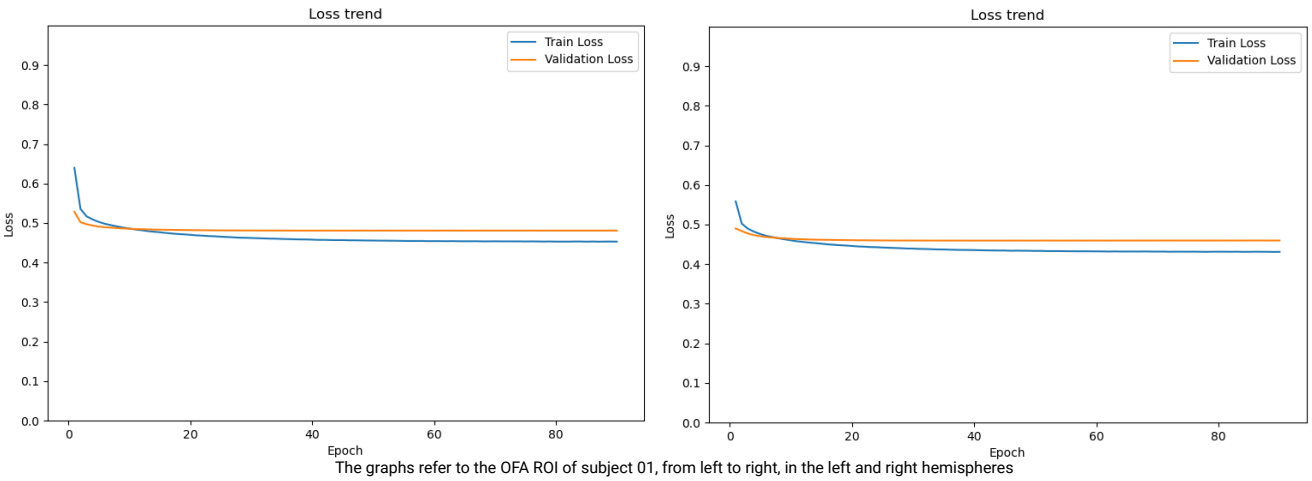
The model architecture that yielded the best results was then the following:

6 1D convolutional layers alternated by BatchNorm1d and LeakyReLU, with the exception that after the second and fourth blocks we added a 20% dropout.

The best hyperparameters configuration we could find was the following:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	7.5e-6	0.75	5 epochs	0.9	0.999	1e-8	1e-2

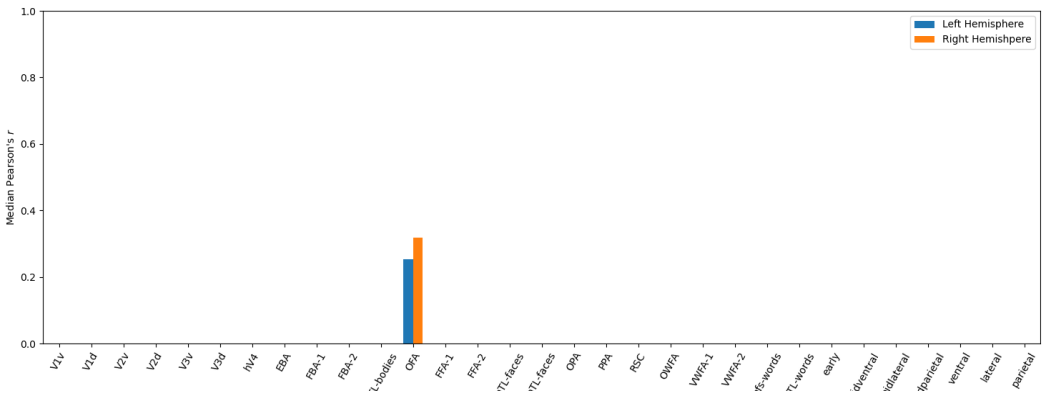
This time, unlike the previously mentioned attempt in this experiment, we managed to get slightly better results compared to the original model:



The graphs refer to the OFA ROI of subject 01, from left to right, in the left and right hemispheres

The validation loss was slightly better than the original model, although the biggest change the regularization bringed was the increase in train loss. This means the model was not overfitting anymore, but was still struggling to generalize properly.

Another good aspect of this new architecture was that, in correlation score for the ROIs it was trained on, it almost managed to get as high as the best model we had so far (CLIP-Simple), being in about 0,01 distance from it. For example, for subject 01:



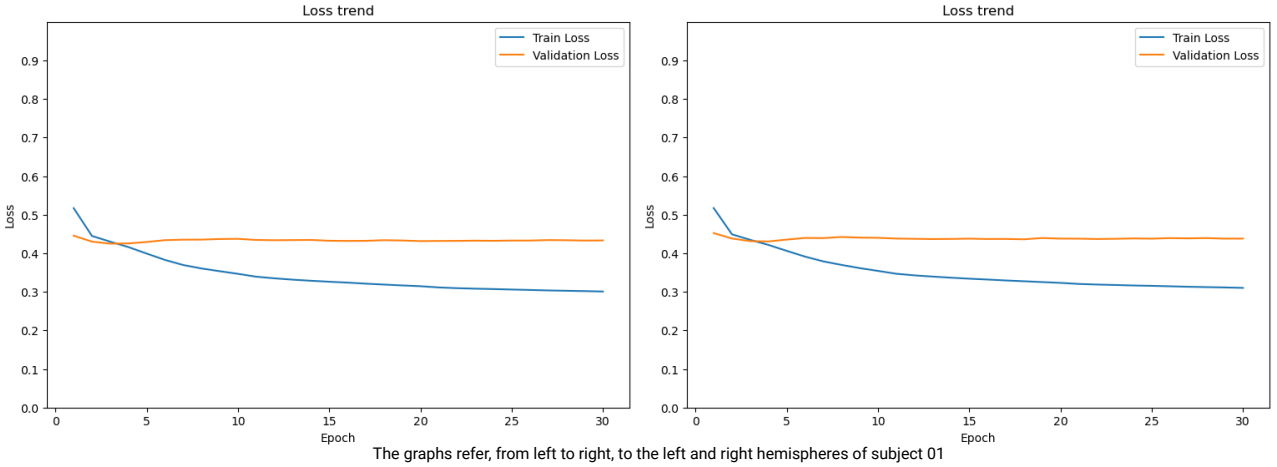
CLIP-Resblocks

To try and further improve the results we obtained with the previous experiments by introducing dropout, since the model was starting to get deep, we also attempted to introduce some residual connection blocks in the model architecture.

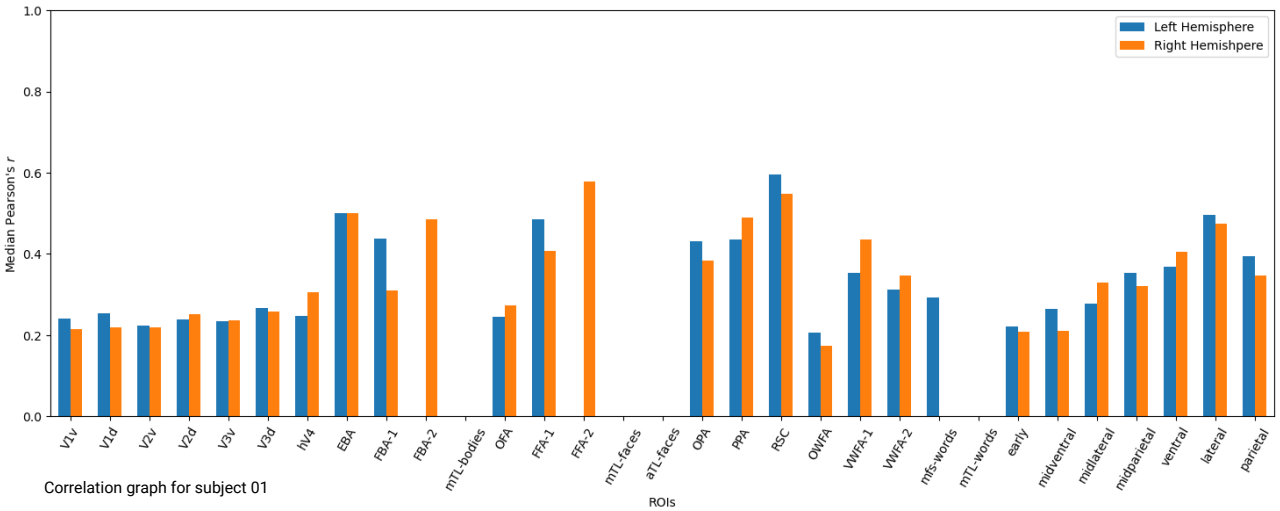
For the first architecture explained in the CLIP-Dropout experiment, the one based on CLIP-DNN, we added 2 residual blocks. We defined a residual block as follows: Linear, BatchNorm1d, LeakyReLU, Linear, BatchNorm1d and LeakyReLU. Each time a residual block was present, we added element-wise its output with its input.

We kept the same 5% dropout in the other non-residual blocks. To increase the capacity of the model, we increased the size of the layer at each non-residual block by a factor of 1.25.

Unfortunately, our best configuration with this architecture still performed worse than the original one, although it did overfit less, similarly to the architecture with dropout only:



The graphs refer, from left to right, to the left and right hemispheres of subject 01

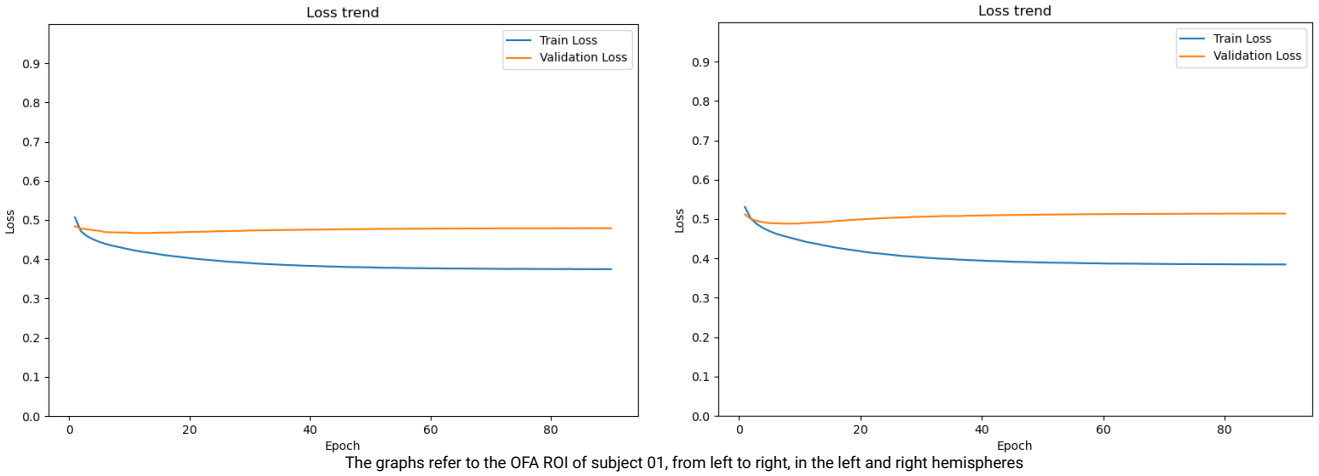


Correlation graph for subject 01

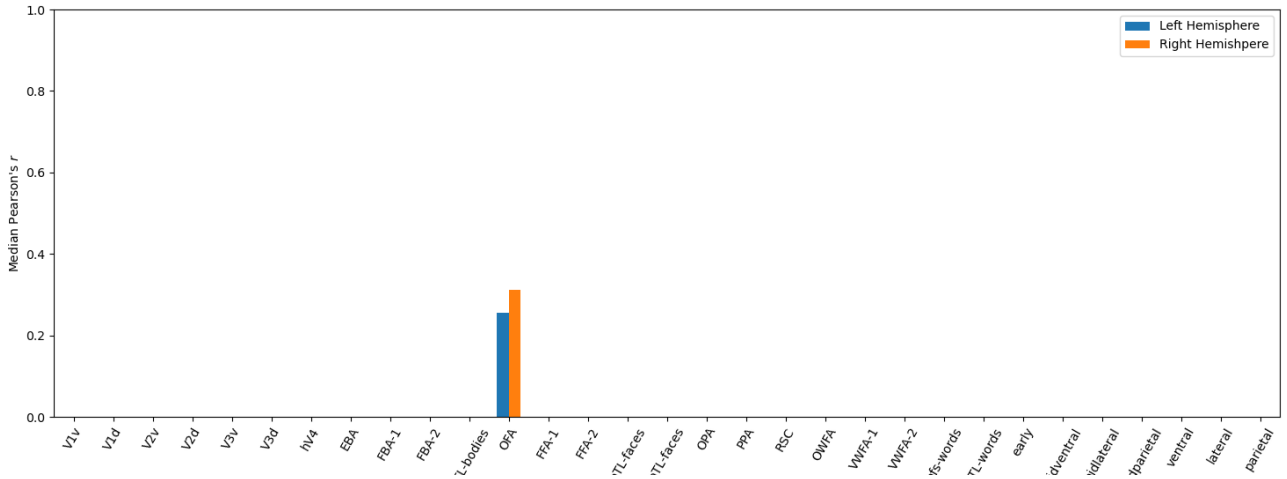
Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	5e-5	0.5	10 epochs	0.9	0.999	1e-8	1e-1

The application of residual blocks to the second architecture explained in the CLIP-Dropout model, the one based on CLIP-Conv1d, did not increase performance.
We tested several hyperparameters configuration, and found out the best results were given by this one:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	1.5e-5	0.75	5 epochs	0.9	0.999	1e-8	1e-2



The graphs refer to the OFA ROI of subject 01, from left to right, in the left and right hemispheres



Above, the correlation graph for the ROI of subject 01 the model was trained on.
The correlation scores were about the same, along with the minimum values of both train and validation losses. However, this model was more stable, which can be seen by the loss trends: the curves are much more smooth and less jiggling.
This is most likely due to the residual blocks allowing a better propagation of the gradient from the last layers, the ones closer to the output, to the first layers, the ones closer to the input. Like its predecessor, it was one of the closest models to our best one so far, but didn't quite reach it yet.

CLIP-ROI

Since all our further attempts to improve the performance of our best model, the one described in the CLIP-Simple experiment, did not have success, we tried a different approach by going back to the starting point and trying to focus more on the prediction of the best model to improve its performance.

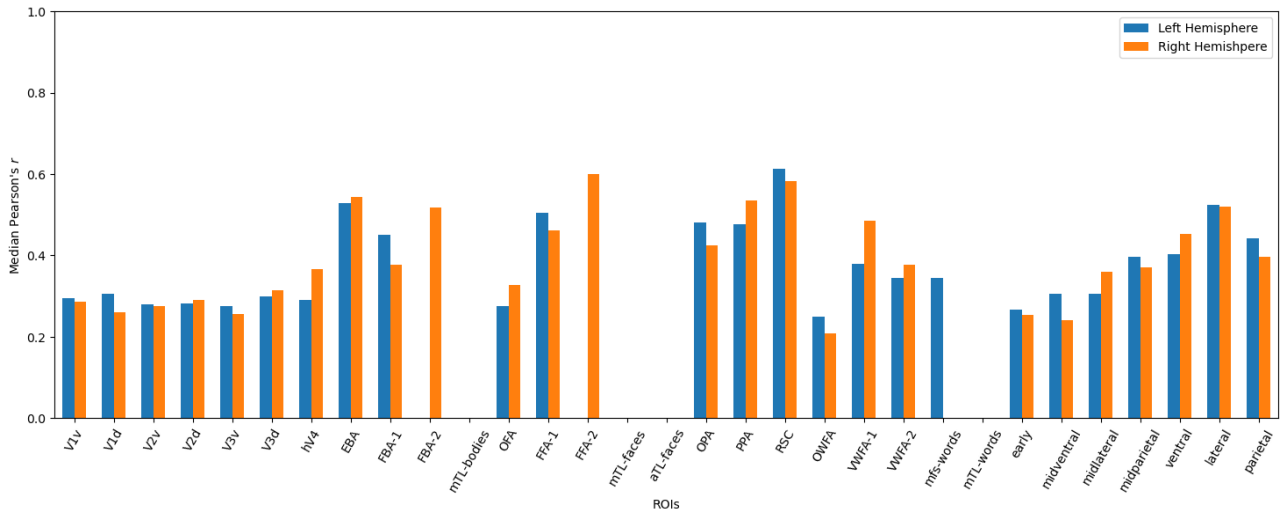
In practice, what we did was apply the same model architecture as our best model, a single linear layer with input CLIP embeddings, but this time it would predict all vertex activations of a single ROI at a time: input CLIP, output N activations, with N the size of the ROI.

Although we weren't expecting much, this model did improve performance compared to our previous best one. Not by a wide margin, but still an improvement in correlation scores across the board of more than 0.01.

The hyperparameters used were the following ones:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	1e-4	0.75	10 epochs	0.9	0.999	1e-8	1e-2

There were instead the correlation scores for subject 01:



As mentioned before, these correlations were higher compared to the previous best model by around 0.01.

An important note is that, although generally, if we look at the loss trends for each ROI, they are mostly higher than the loss trends we saw for the previous best model, it must be considered that the previous best model loss was averaged across ROIs due to the fact that it was predicting whole brain activation.

CLIP-Comb

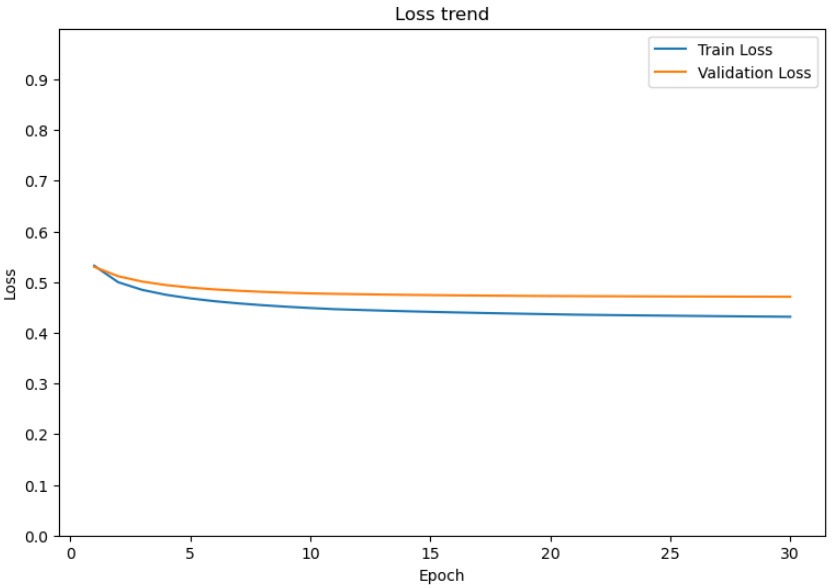
In light of the previous experiment results, we tried to combine that model with each of the attempts from the CLIP-Conv1d, CLIP-Dropout, and CLIP-Resblocks experiments that were also taking as input the positional encodings and had the lowest train loss.

The hope was, since the CLIP-ROI experiment showed a high correlation between validation and train loss, that by combining two models, the final model would learn how to best give importance to which model and when, allowing it to lower the train loss while keeping the validation loss close to it.

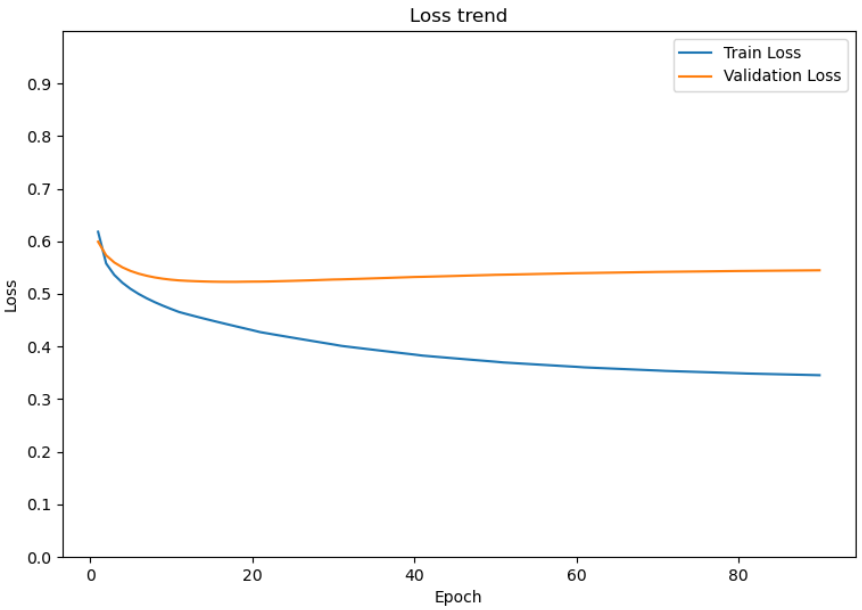
To combine the CLIP-ROI experiment model with any of the other ones, we opted for a model that would take as input CLIP image embeddings and would:

1. (CLIP-ROI) Feed that to a single linear layer, which would output N vertices activation predictions, where N is the size of the current ROI.
2. (Others) Feed that input along with positional encodings to a sequence of layers that would resemble the other model we were combining and output a single vertex activation prediction. To this submodule, we would feed a batch of all vertex inputs, such that the shape of the output could be combined with the output from the other submodule.
3. The results from both submodels would be combined through an element-wise sum.

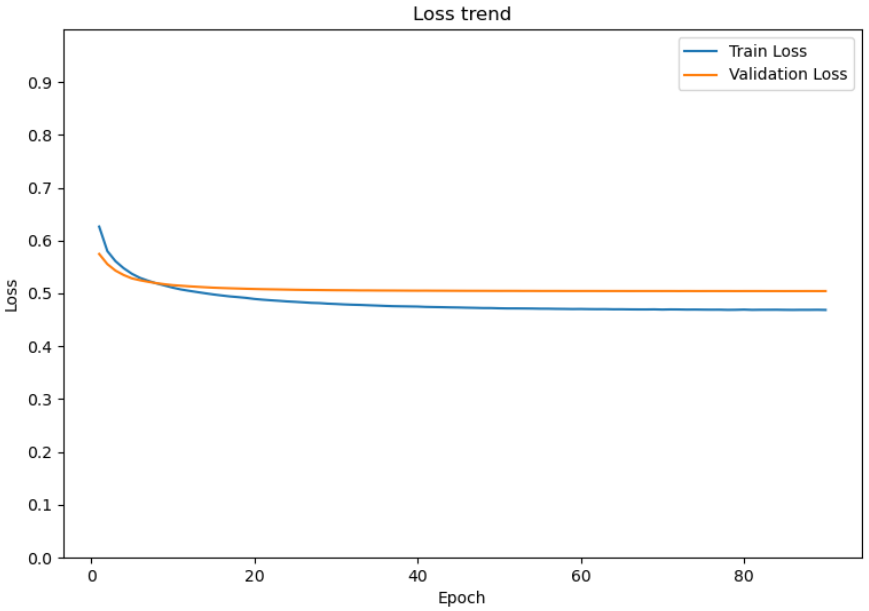
Regrettably, the outcome of this experiment did not align with our initial hopes. Each of the combined models exhibited higher validation loss and consequently lower correlation scores. Nevertheless, there was a silver lining in the form of significantly reduced train loss compared to the CLIP-ROI model. However, this outcome was expected, considering the increased complexity of the models involved in the combination.



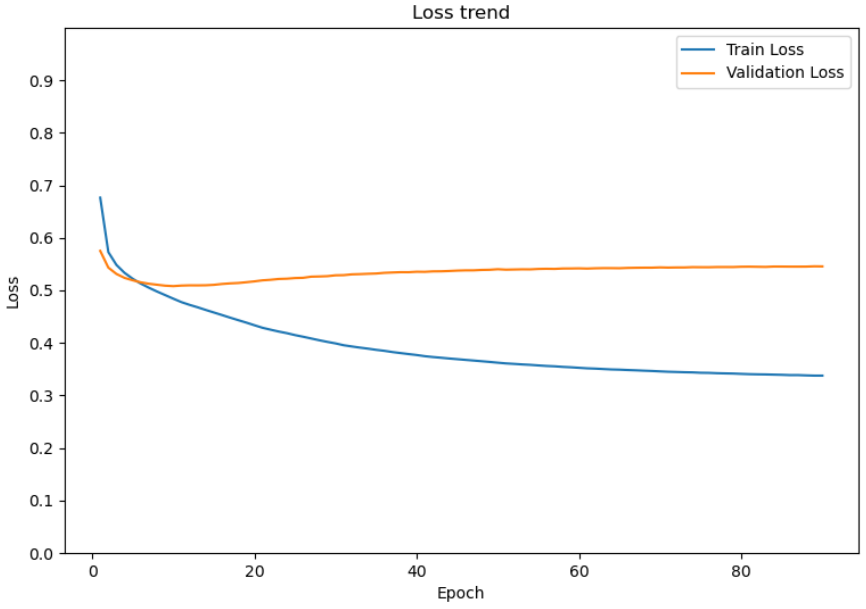
For comparison with the next graphs, the one on the left is the loss trends graph for the CLIP-ROI model alone, for the same subject, same ROI (OFA) and same hemisphere (left) as the other graphs in the next page refer to.



This is an example of the loss trends for CLIP-ROI + CLIP-Conv1d. The train loss decreased significantly, but the validation loss went up compared to only CLIP-ROI. The model was, however, more stable compared to only CLIP-Conv1d.



To the left, one of the graphs for the loss trends of CLIP-ROI + CLIP-Dropout. As before, the model did get slightly better than CLIP-Dropout alone, but was a little worse than just CLIP-ROI.



An example of the loss trends for the last combination, CLIP-ROI + CLIP-Resblocks.
This time, the model had a train loss lower than both models it's composed of, however the validation loss was still higher than CLIP-ROI only.

Summary

The Algonauts Challenge 2023 presented an intriguing task, requiring the development of a sophisticated model capable of accurately predicting brain activation in specific regions of interest. During the time we had at our disposal, we thoroughly explored various approaches to generate these predictions. To enhance the capabilities of our own models, we utilized three different large-scale models to process the input images and generate image embeddings. These embeddings allowed us to leverage the extensive and much larger datasets on which these models were trained, benefiting from their exceptional capacity to generalize and produce meaningful and semantically rich embeddings.
Among the selected large-scale models, namely AlexNet, CLIP, and SAM, it quickly became evident that CLIP exhibited the most promising results in terms of both performance and training speed. Armed with this knowledge, we set out to surpass the baseline provided by the Challenge organizers, which utilized AlexNet features fitted into a linear regression model to predict brain activation. By employing CLIP-generated image embeddings and constructing a slightly more intricate model comprising a single linear layer, we swiftly surpassed the baseline, effectively establishing it as our new reference point.
Nevertheless, further improving our initial results proved to be a formidable challenge, as alternative models we explored consistently delivered subpar performance in terms of both training speed and correlation scores, with only a select few models managing to approximate our own baseline.
Our most relevant attempts included, but were not limited to: fine-tuning hyperparameters; providing additional information to the input through carefully created positional encodings of the current activation to predict; separating the predictions into single ROIs or single vertices, increasing models complexity; integrating and fine-tuning regularization techniques such as dropout; addressing a possible problem of vanishing gradients by adding residual connection layers; and combining the predictions from our top-performing models in terms of train loss with the most successful model in terms of validation loss.
With all of this effort, our research and time ended up with a slightly better model than our own baseline. This model retained the same architecture but was refined to predict one ROI activation at a time. While the improvements may not have been as substantial as initially envisioned, the rigorous exploration of methodologies and techniques allowed us to gain many insights into the intricate task of predicting brain activation, allowing us to deepen our knowledge of both AI and human brain behavior prediction tasks.

Best model and results

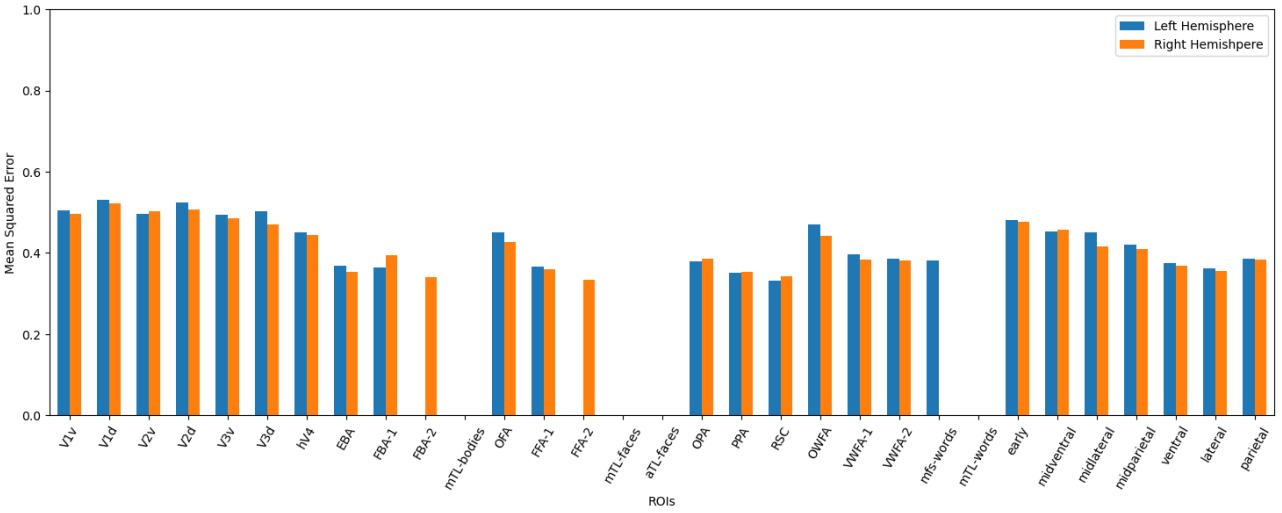
The best model we could devise was a model consisting of a single fully connected linear layer focused on predicting one whole ROI at a time for each hemisphere for each subject. We found out that predicting one ROI at a time improved performance compared to predicting whole brain activity, which indicates that most likely inter-ROI interactions play a less important role compared to intra-ROI interactions.

As previously mentioned, our best architecture was trained to specifically predict different ROIs for different subjects in different hemispheres. For this reason, we created several models with the same architecture, one for each ROI of each hemisphere of each subject. The available data was divided into two partitions: a training one with 90% of the data and a validation one with the remaining 10%.

Each image in the dataset had its corresponding fMRI data target, and each of our models took one image as input and predicted specifically the portion of the fMRI data relative to its subject, hemisphere, and ROI of scope.

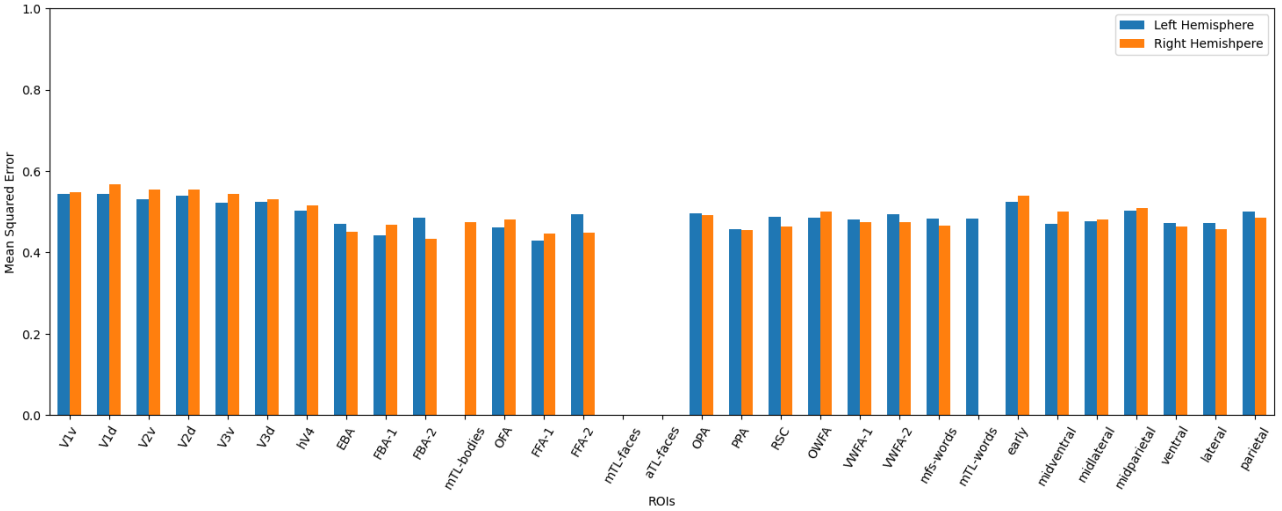
We found out that generalizing across subjects was a particularly hard task, and for this reason, we obtained the best results when models were trained to predict brain responses for a specific subject. In total, we had about $8 \times 2 \times 26$ different models because there were 8 subjects, 2 hemispheres for each subject, and around 26 ROIs for each hemisphere. It's important to note that not all ROIs exist in both hemispheres and in all subjects.

The MSE loss for most subjects was similar to the one reported here for subject 01:

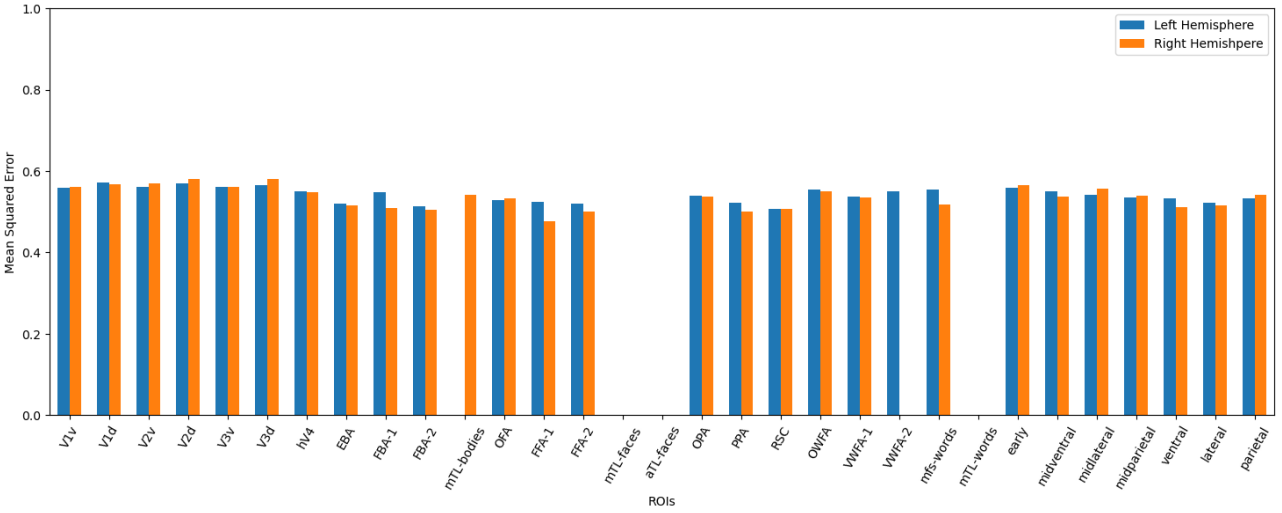


However, as was to be expected, subjects 06 and 08 had significantly lower performance, which translated into higher MSE loss values due to the fact that they had missing fMRI data.

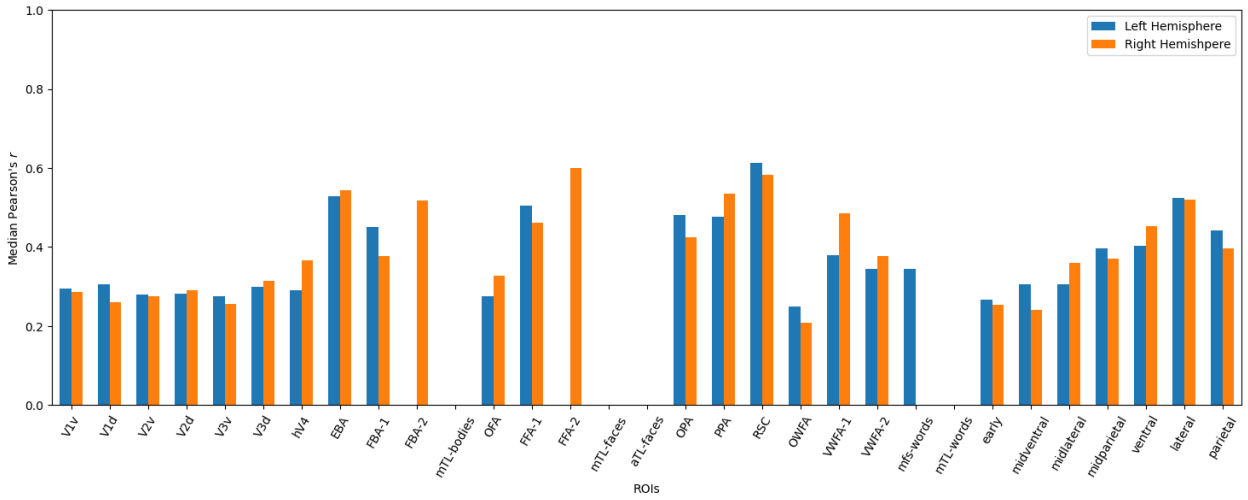
The graph below shows the MSE loss values for subject 06:



This other graph below illustrates instead the MSE loss values for subject 08:

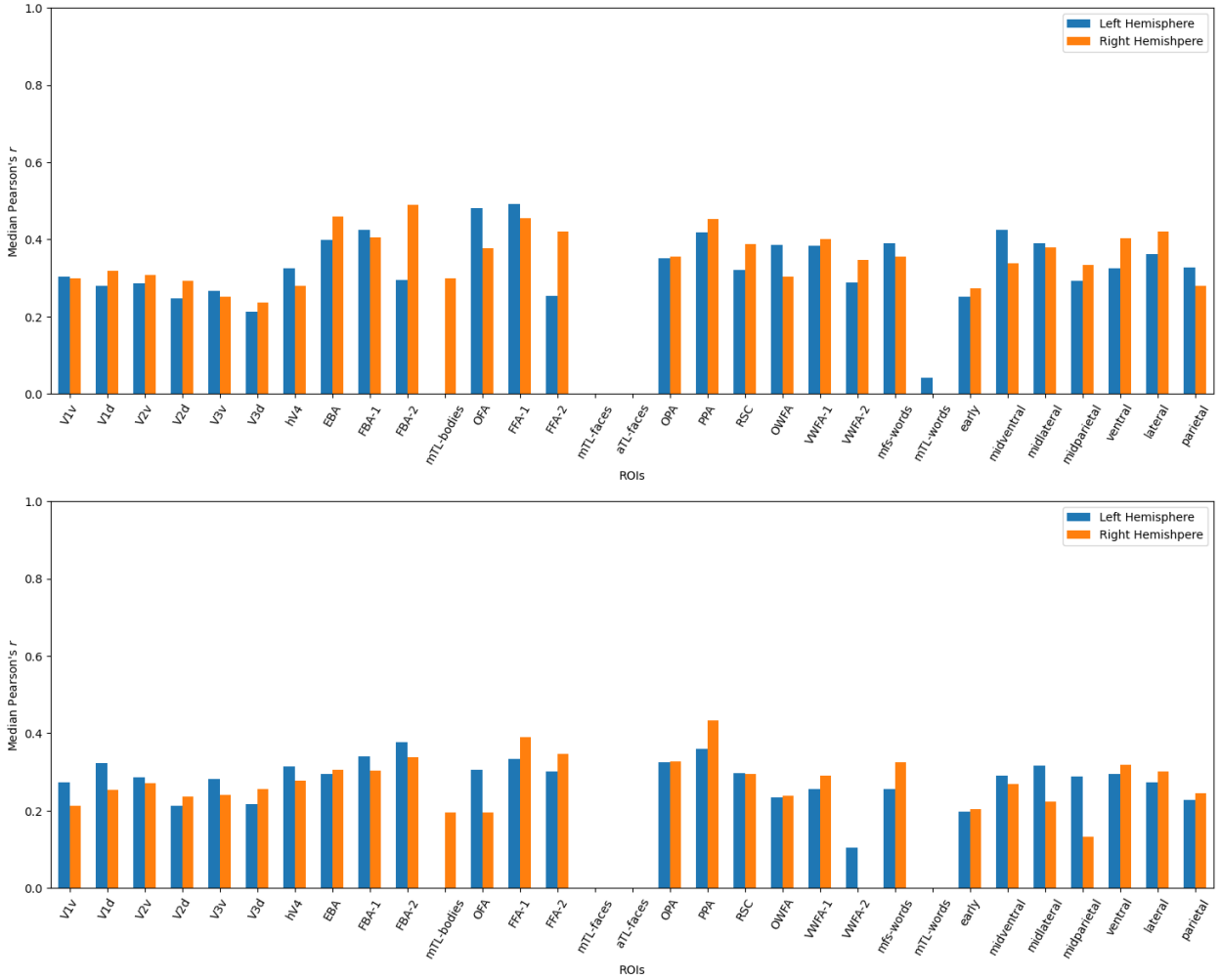


The correlation scores, which measure the relationship between predicted brain activity and actual brain activity, exhibit a significant association with the corresponding loss values. This correlation is precisely what one would anticipate: when the loss values are higher, the quality of predictions diminishes, resulting in a lower correlation between the predicted and actual brain activity.



The graph above refers to the correlation scores for subject 01. It's plain to see that the ROIs with the highest correlation scores are the ones for which the MSE loss value is the lowest in the previous graph showing the loss values for subject 01; vice versa, the ROIs with the highest loss values are the ones with the lowest correlation scores.

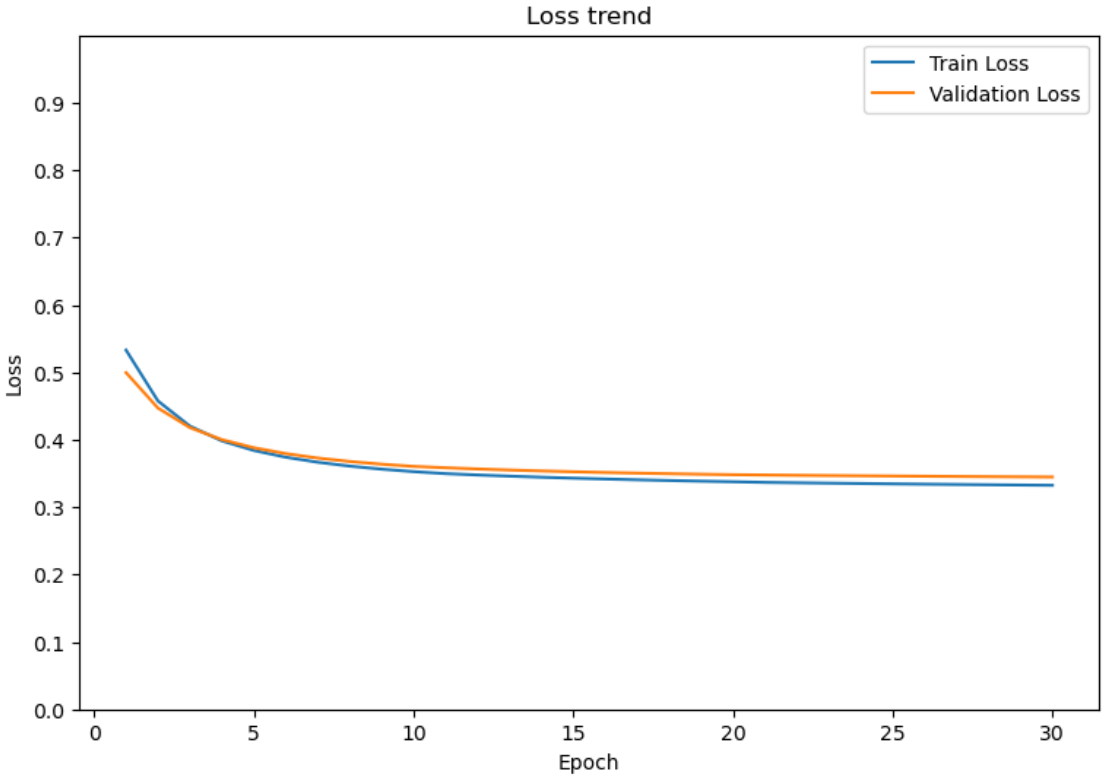
Looking at the correlation graphs for subject 06 and subject 08:



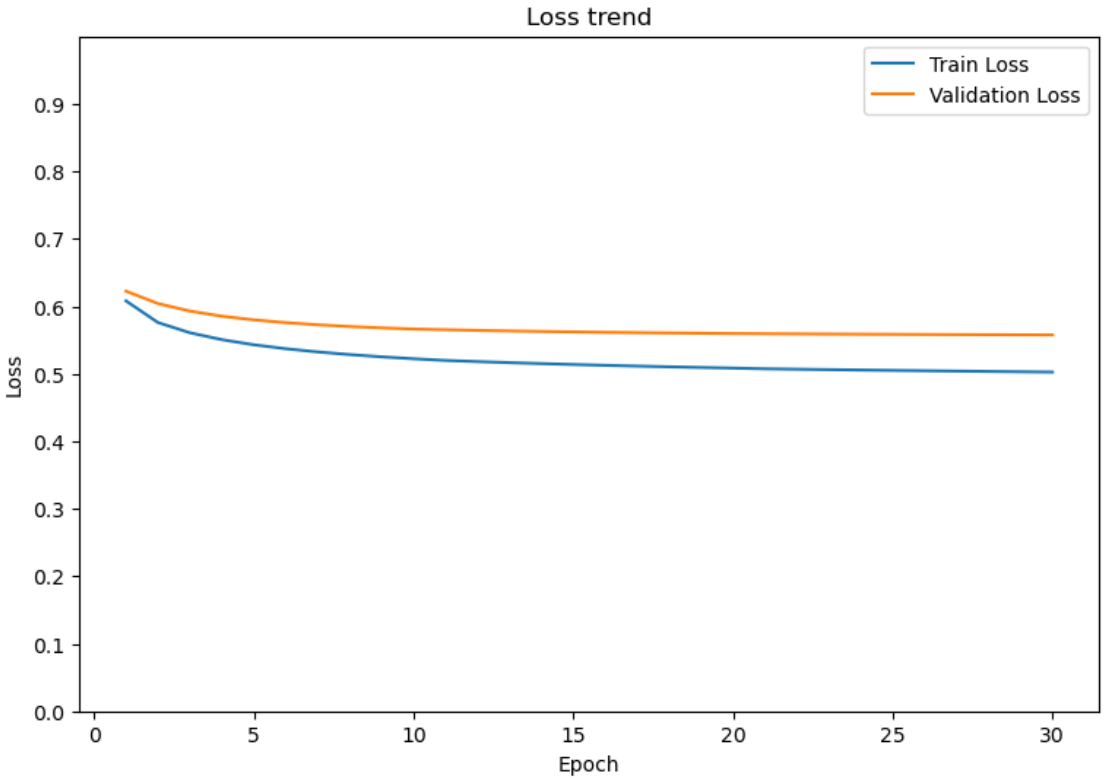
It's evident how the correlation scores are lower than other subjects, for example the reference subject 01, which fits the fact that their loss values are generally higher.

We trained these models for 30 epochs, as we found that further epochs had diminished returns.

For instance, below the graph of the loss trends for subject 01, left hemisphere, RSC region, one of the lowest in terms of MSE loss:



Instead this other one refers to the same subject and hemisphere, but for the V1d ROI:



The situation is similar across all the models under consideration. Upon analyzing the loss trends graph for each model, it becomes evident that both the train loss and validation loss exhibit a consistent pattern. In particular, during the initial 10 epochs, there is a noticeable decline in both the train and validation loss, indicating that the models are improving in terms of their predictive performance.
However, as the training progresses beyond the initial epochs, the loss plots begin to display a more stationary behavior. This suggests that the rate of improvement gradually slows down, and the models reach a point of diminishing returns in terms of reducing the loss. In other words, while there may still be slight fluctuations in the loss values, they tend to stabilize around a certain level without significant further reduction.

The following hyperparameters were used as we tested multiple configurations and found out this one to be the one yielding the best results:

Optimizer	Learning Rate	LR decay	LR step	β_1	β_2	ε	Weight decay
AdamW	1e-4	0.75	10 epochs	0.9	0.999	1e-8	1e-2

Despite achieving our best result so far, we believe that there is still significant room for improvement: none of the ROIs in any of the subjects managed to surpass a correlation score of 0.7, which is considered indicative of a strong correlation between our predictions and the actual brain activation, and certain ROIs obtained correlation scores as low as 0.3, falling below the threshold that defines a weak correlation.
Interestingly, the most successful models we employed were surprisingly simple.
Although this simplicity may have contributed to their superior performance, we anticipate that utilizing more intricate models could potentially yield even better results. However, our attempts to construct such complex models did not yield any improvements over the performance achieved by the simpler architecture described here.
Overall, while we acknowledge that our current findings represent our best efforts thus far, we remain optimistic about the possibility of further advancements. We believe that exploring more sophisticated models and refining our methodologies will enable us to unlock the true potential of our research and establish stronger correlations between our predictions and actual brain activation patterns.

Future developments and possible improvements

Despite the fact that we managed to surpass the Algonauts Challenge 2023 baseline model, we are not quite satisfied, as we improved by a thin margin and only in some ROIs, while a few others saw a decrease in correlation.

We hypothesize this may be due to the embeddings we used compared to the baseline. We used CLIP embeddings that are rich in semantics and indeed are closely related to textual information, while AlexNet embeddings better capture the visual and spatial features of images. To further strengthen this idea, we noticed that the ROIs that saw a decrease in their correlation score were the ones in the class called "prf-visualrois", or "Early retinotopic visual regions". Combining CLIP and AlexNet embeddings could help.

Something that surprised us was the fact that our simplest models were the ones that yielded the best results. Generally speaking, deeper and more complex models should be able to learn much more complex functions and fit the data better. This is especially true when considering that our simplest models did not even introduce any non-linearity, although it must be noted that non-linearity is introduced beforehand thanks to transferring learning from pre-trained large-scale models. Most likely, the advantage that allowed simpler models to have the best of more complex models was that the latter would, most of the time, overfit to the training data instead of properly generalizing. We did try several configurations and regularization techniques, but to no avail: either the models would keep overfitting or generally have worse performance. Along the same line, we struggled to get models based on SAM embeddings stable, and even our best result showed only a more or less stable training loss, while the validation loss would still jiggle considerably, albeit not as much as in initial attempts. We believe it could be due to the much more complex model architecture we had to use to process SAM embeddings or to the Visual Transformer used by SAM being too fine-tuned on segmentation tasks.

We noticed our positional encoding was, while good, not optimal for the smallest ROIs. This could be improved by increasing the factor by which the bounding box is expanded.

On a final note, while it’s true that the dataset available for this competition was the largest suitable dataset to date, it’s still far from being considered an exceptionally large dataset. For comparison, the ImageNet dataset contains around 14 million labeled images, CLIP was trained on around 15 million image-text pairs, and SAM was trained on 1 billion images, while the NSD dataset only has 73 thousand images. Improving results could also be achieved by increasing the available training data.

Conclusion

After devoting significant time and effort to this project, we made the decision to submit the predictions generated by our best models to the challenge, eagerly awaiting the final results. Surpassing the challenge baseline was a significant achievement, but we remain confident that there is still room for improvement. Our extensive experimentation substantiated this belief, as we discovered that more complex architectures were unable to outperform simpler ones, indicating a tendency to overfit.

Throughout our journey, we immersed ourselves in comprehensive research on brain activity and familiarized ourselves with the structure of the NSD dataset. Initially, we employed straightforward models to predict brain activity. However, as time progressed, we gradually introduced more intricate models that made use of the available data and embeddings generated by large-scale pre-trained models, exploring diverse avenues for processing them.

Developing our models presented a range of challenges. Notably, we encountered difficulties in effectively addressing regularization and overfitting in our most complex models. In particular, models based on SAM embeddings struggled to maintain stability, adding an extra layer of complexity to the development process.

After countless iterations and attempts, we discovered that employing multiple models yielded the best results. Specifically, we utilized a separate model for each ROI in each hemisphere of every subject. These models shared a common architecture consisting of a single fully connected linear layer. The input to this layer comprised image embeddings generated by CLIP, and the output provided predictions for the activation levels of each vertex within the ROI.

Ultimately, despite achieving our best results, we recognize that there is still ample opportunity for improvement. Even though our models achieved a reasonably strong correlation with the available data, they fell just short of establishing a robust association. This realization motivates us to continue refining our approach and pushing the boundaries of our predictions.

Bibliography

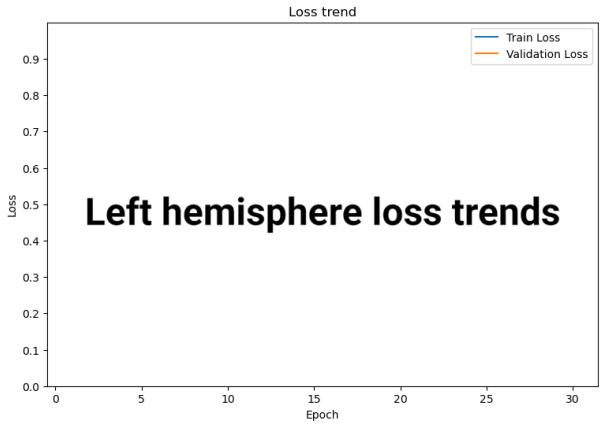
1. Allen, Emily J., et al. 'A Massive 7T fMRI Dataset to Bridge Cognitive Neuroscience and Artificial Intelligence'. *Nature Neuroscience*, vol. 25, no. 1, Jan. 2022, pp. 116–26. *www.nature.com*, https://doi.org/10.1038/s41593-021-00962-x.
2. Cichy, Radoslaw Martin, et al. *The Algonauts Project: A Platform for Communication between the Sciences of Biological and Artificial Intelligence*. arXiv, 14 May 2019. *arXiv.org*, https://doi.org/10.48550/arXiv.1905.05675.
3. Dosovitskiy, Alexey, et al. *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv, 3 June 2021. *arXiv.org*, https://doi.org/10.48550/arXiv.2010.11929.
4. Gifford, A. T., et al. *The Algonauts Project 2023 Challenge: How the Human Brain Makes Sense of Natural Scenes*. arXiv, 11 July 2023. *arXiv.org*, https://doi.org/10.48550/arXiv.2301.03198.
5. He, Kaiming, Xiangyu Zhang, et al. 'Deep Residual Learning for Image Recognition'. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 770–78. *DOI.org (Crossref)*, https://doi.org/10.1109/CVPR.2016.90.
6. He, Kaiming, Xinlei Chen, et al. 'Masked Autoencoders Are Scalable Vision Learners'. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 15979–88. *IEEE Xplore*, https://doi.org/10.1109/CVPR52688.2022.01553.
7. Ioffe, Sergey, and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv, 2 Mar. 2015. *arXiv.org*, https://doi.org/10.48550/arXiv.1502.03167.
8. Kirillov, Alexander, et al. *Segment Anything*. arXiv, 5 Apr. 2023. *arXiv.org*, https://doi.org/10.48550/arXiv.2304.02643.
9. Krizhevsky, Alex, et al. 'ImageNet Classification with Deep Convolutional Neural Networks'. *Communications of the ACM*, vol. 60, no. 6, May 2017, pp. 84–90. *DOI.org (Crossref)*, https://doi.org/10.1145/3065386.
10. Radford, Alec, et al. *Learning Transferable Visual Models From Natural Language Supervision*. arXiv, 26 Feb. 2021. *arXiv.org*, https://doi.org/10.48550/arXiv.2103.00020.

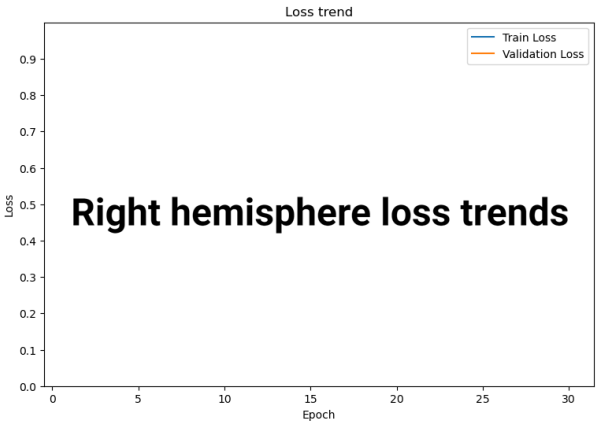
11. Takagi, Yu, and Shinji Nishimoto. *High-Resolution Image Reconstruction with Latent Diffusion Models from Human Brain Activity*. bioRxiv, 11 Mar. 2023. *bioRxiv*, https://doi.org/10.1101/2022.11.18.517004.
12. Vaswani, Ashish, et al. *Attention Is All You Need*. arXiv, 5 Dec. 2017. *arXiv.org*, https://doi.org/10.48550/arXiv.1706.03762.
13. Yang, Huzheng, et al. *Effective Ensemble of Deep Neural Networks Predicts Neural Responses to Naturalistic Videos*. bioRxiv, 27 Aug. 2021. *bioRxiv*, https://doi.org/10.1101/2021.08.24.457581.
14. Cichy, R. M., et al. *The Algonauts Project 2021 Challenge: How the Human Brain Makes Sense of a World in Motion*. 1, arXiv, 28 Apr. 2021. *arXiv.org*, https://doi.org/10.48550/arXiv.2104.13714.
15. Naselaris, Thomas, et al. 'Encoding and Decoding in fMRI'. *NeuroImage*, vol. 56, no. 2, May 2011, pp. 400–10. *ScienceDirect*, https://doi.org/10.1016/j.neuroimage.2010.07.073.
16. Lin, Tsung-Yi, et al. *Microsoft COCO: Common Objects in Context*. arXiv, 20 Feb. 2015. *arXiv.org*, https://doi.org/10.48550/arXiv.1405.0312.
17. Loshchilov, Ilya, and Frank Hutter. *Decoupled Weight Decay Regularization*. arXiv, 4 Jan. 2019. *arXiv.org*, https://doi.org/10.48550/arXiv.1711.05101.
18. Reddi, Sashank J., et al. *On the Convergence of Adam and Beyond*. 2018. *openreview.net*, https://openreview.net/forum?id=ryQu7f-RZ.

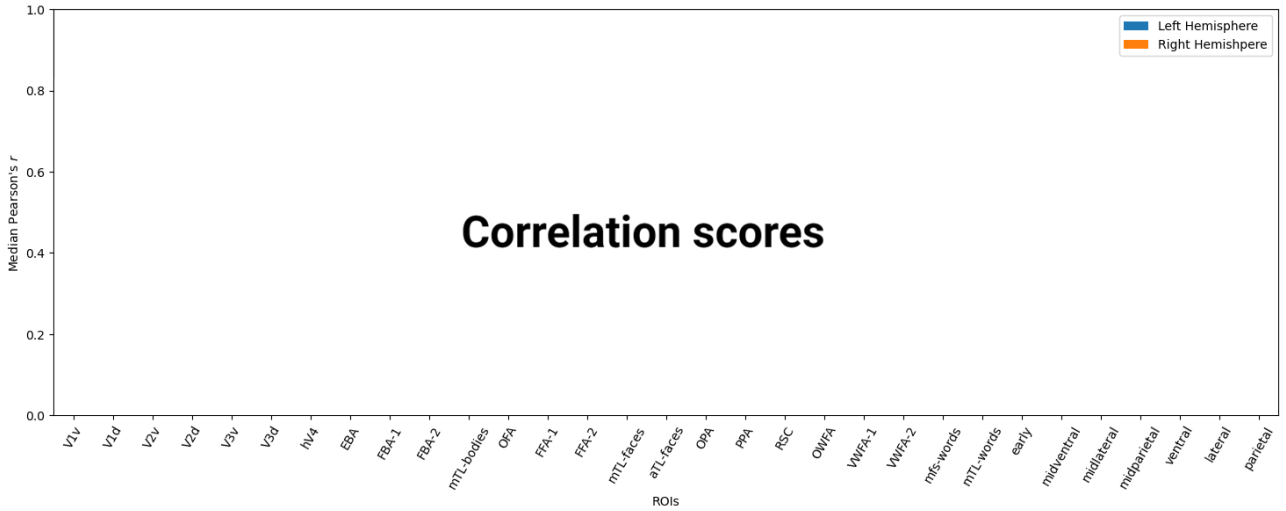
Appendix

Legend

ROI(s)	Overfit	Configuration	Architecture	Loss SX train	Loss SX val	Correlation SX
				Loss DX train	Loss DX val	Correlation DX







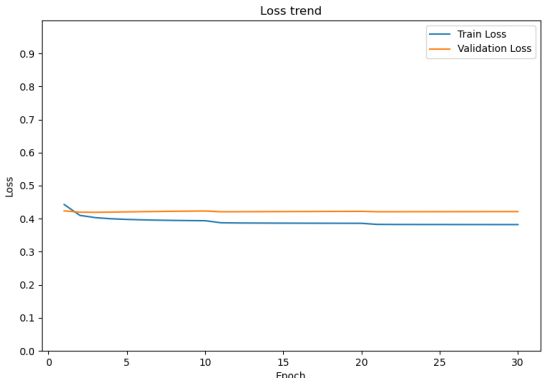
All models reported from next page used AdamW optimizer with default parameters:

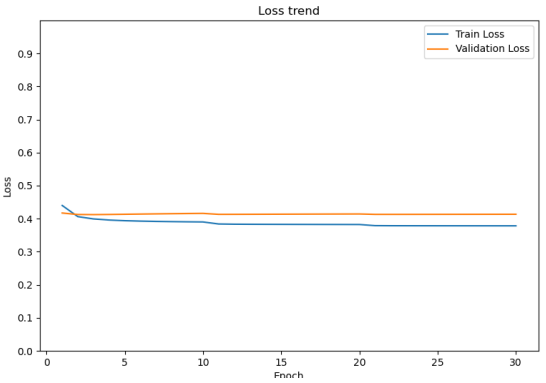
β_1	β_2	ε	Weight decay
0.9	0.999	1e-8	1e-2

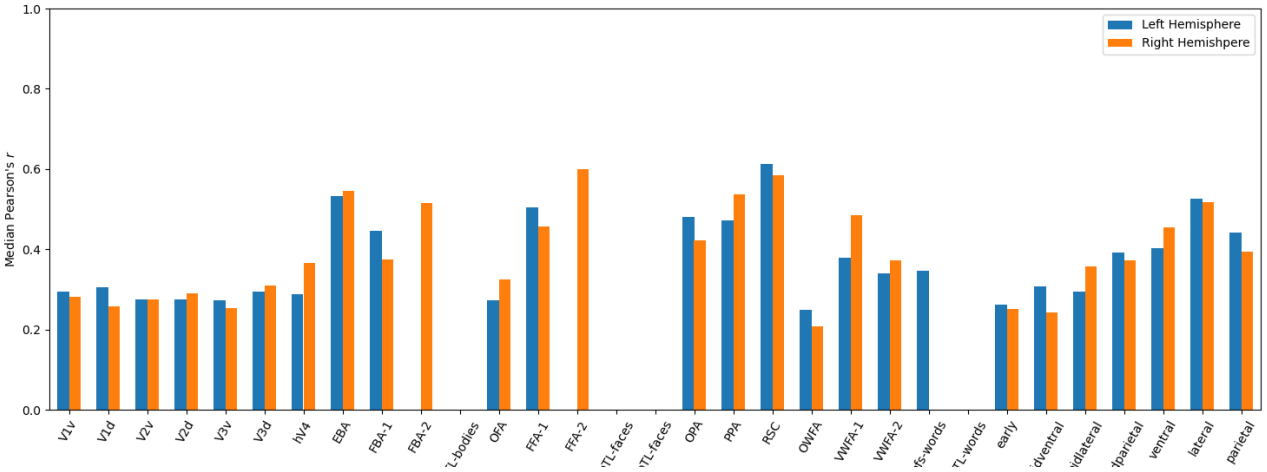
All model metrics refer to subject 01 and to the OFA ROI, unless the model was predicting all ROIs simultaneously; in that case, the minimum loss values are averaged across all ROIs, however the correlation values (not the graph) will still be about the OFA ROI.

CLIP-Simple

All	No	LR: 1e-03 Step: 10 Decay: 0.5	Single linear layer	0.3821	0.4214	0.2726
				0.3784	0.4134	0.3257

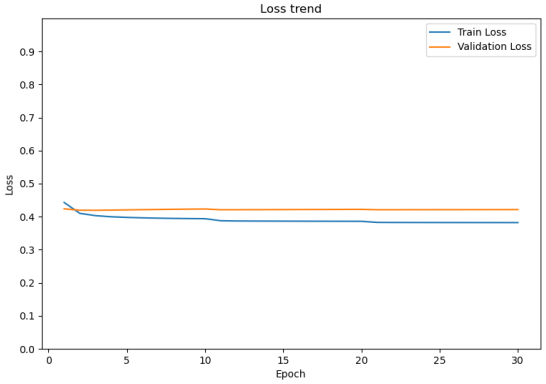


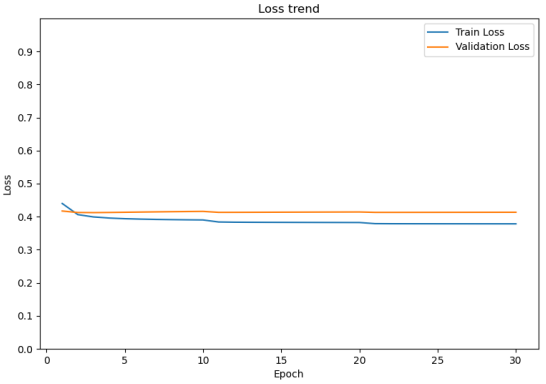


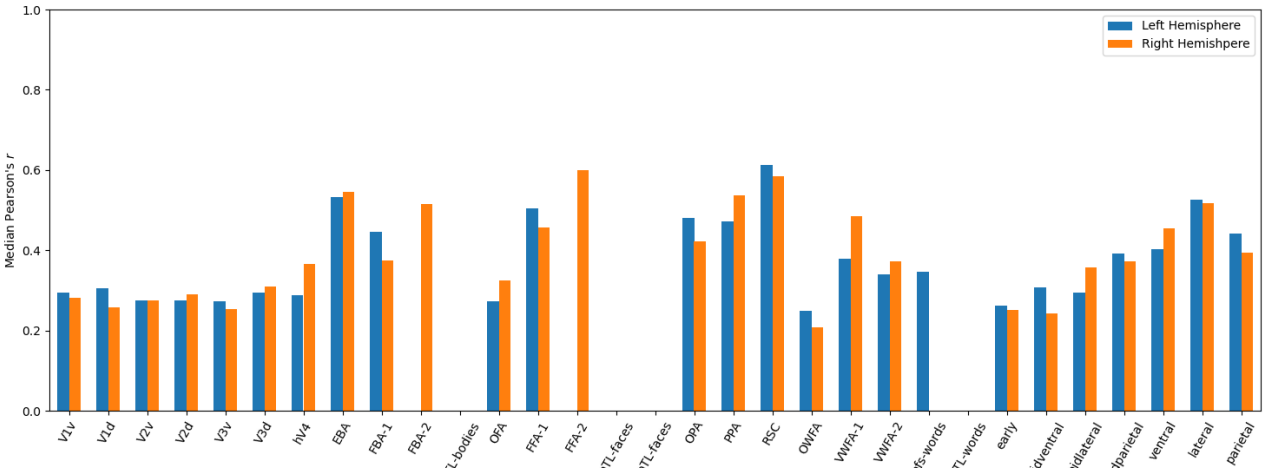


CLIP-Simple-All

All	No	LR: 1e-03 Step: 10 Decay: 0.5	Single linear layer Trained on all subjects	0.4547	0.5004	0.1901
				0.4488	0.4940	0.1744

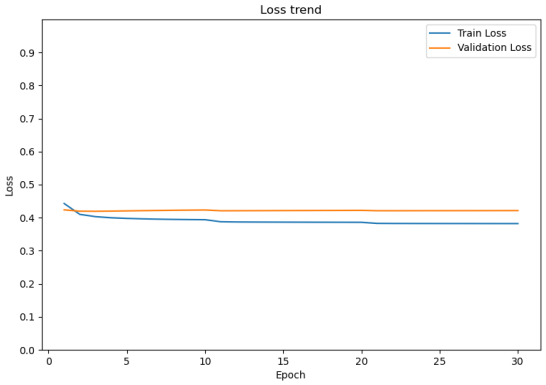


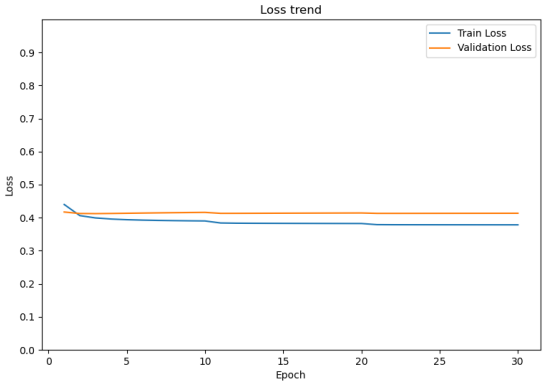


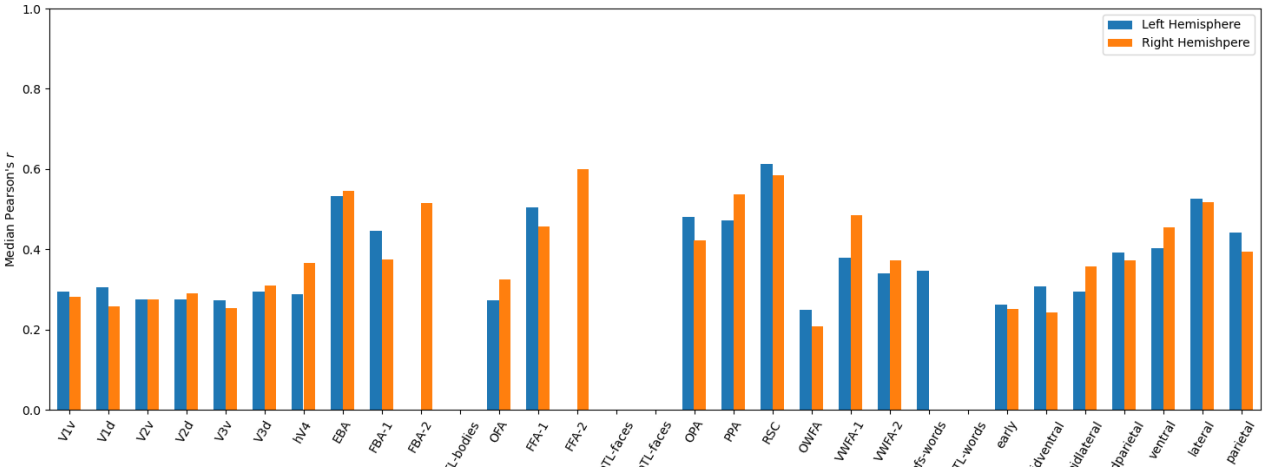


CLIP-DNN

All	Yes	LR: 1e-03 Step: 10 Decay: 0.5	4 linear layers with in between BatchNorm1d and LeakyReLU	0.2105	0.4268	0.2580
				0.2050	0.4245	0.3122

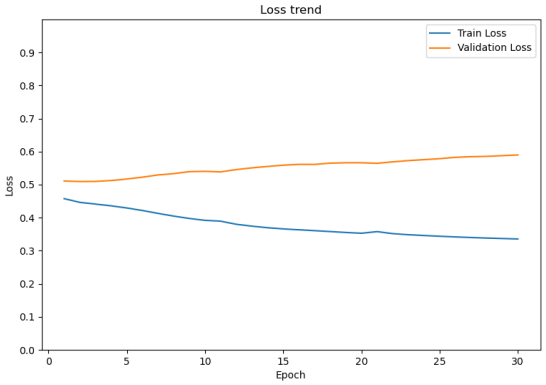


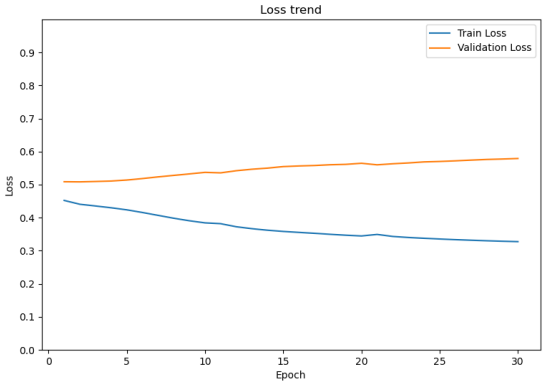


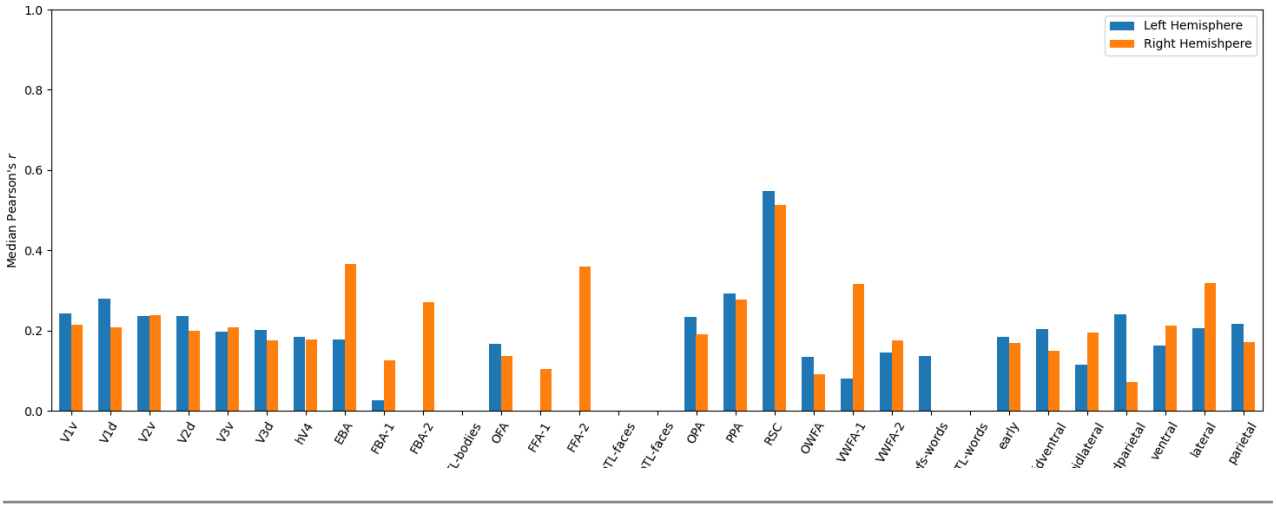


CLIP-DNN-All

All	Yes	LR: 1e-03 Step: 10 Decay: 0.5	4 linear layers with in between BatchNorm1d and LeakyReLU Trained on all subjects	0.3355	0.4937	0.1659
				0.3275	0.5084	0.1371

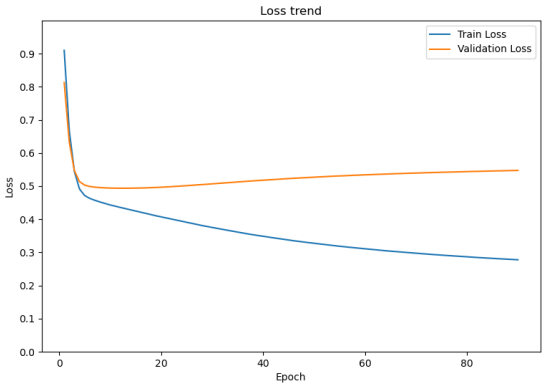


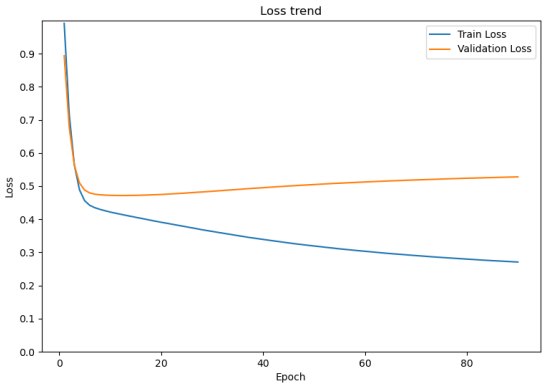


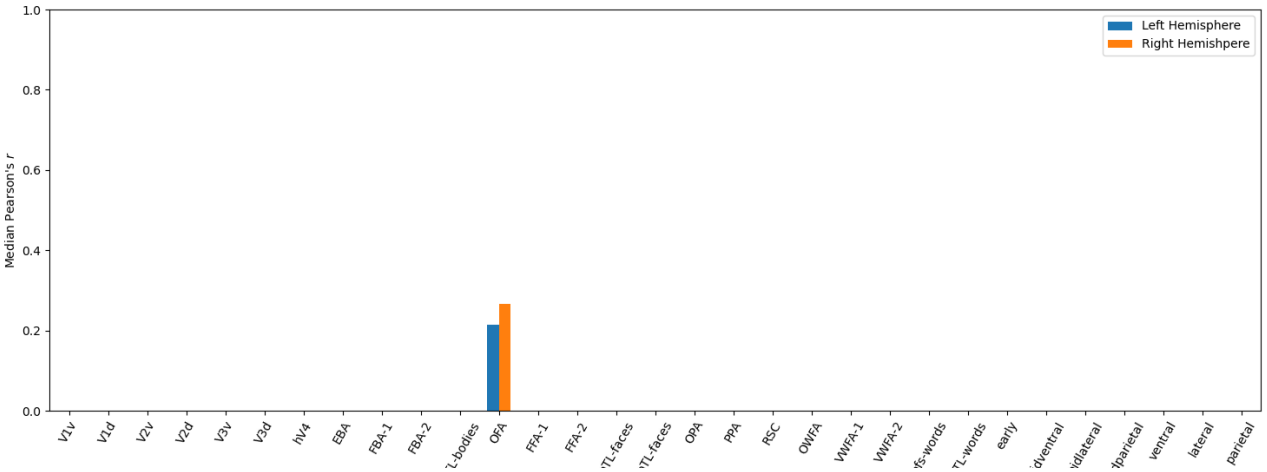


CLIP-Conv1d (First)

Single	Yes	LR: 5e-06 Step: 9 Decay: 0.9	6 Conv1d layers, BatchNorm1d and LeakyReLU, positional attention at each step	0.2776	0.5004	0.2146
				0.2708	0.4717	0.2664

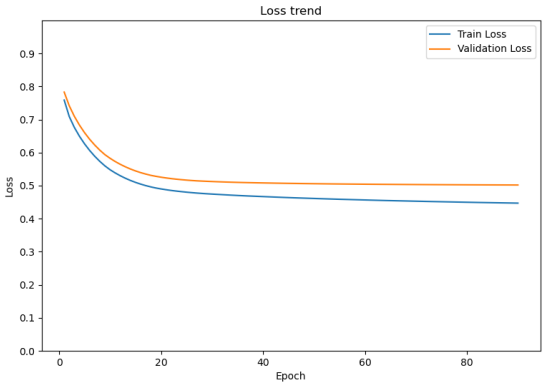


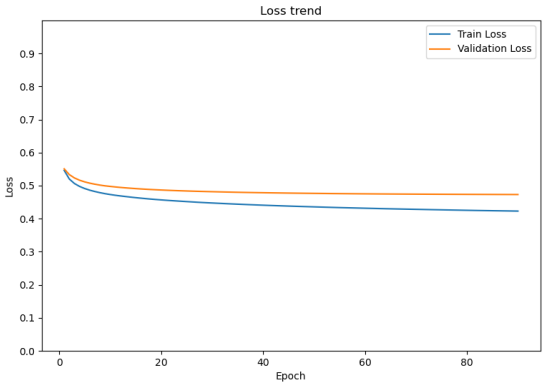


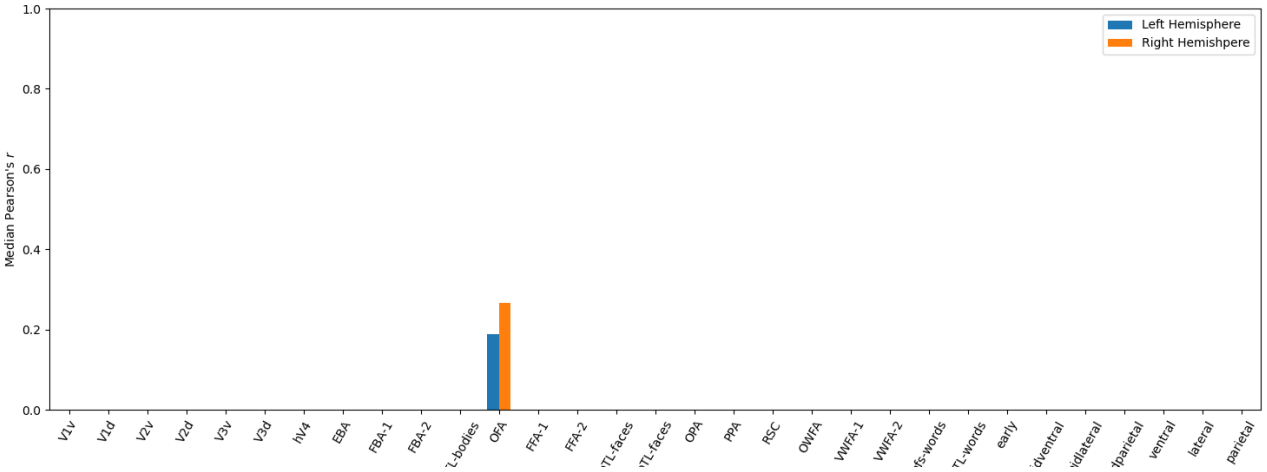


CLIP-Conv1d (Less overfit)

Single	No	LR: 6.25e-07 Step: 9 Decay: 0.9	6 Conv1d layers, BatchNorm1d and LeakyReLU, positional attention at each step	0.4471	0.5020	0.1873
				0.4230	0.4731	0.2661

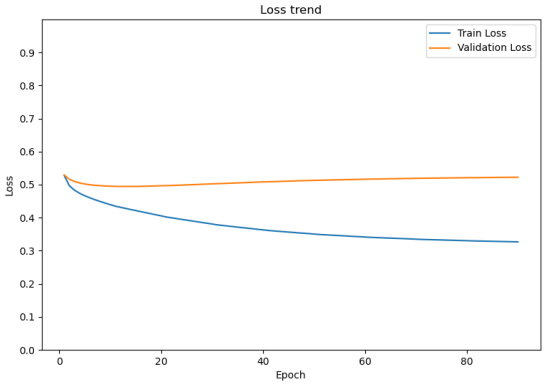


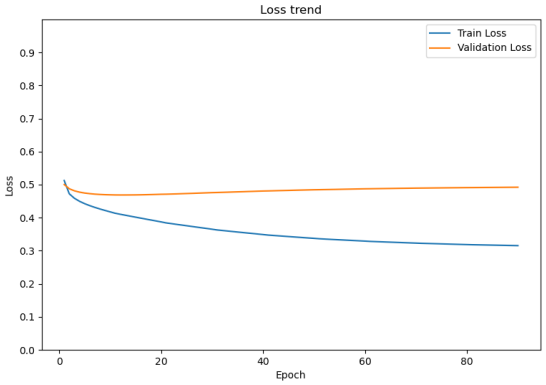


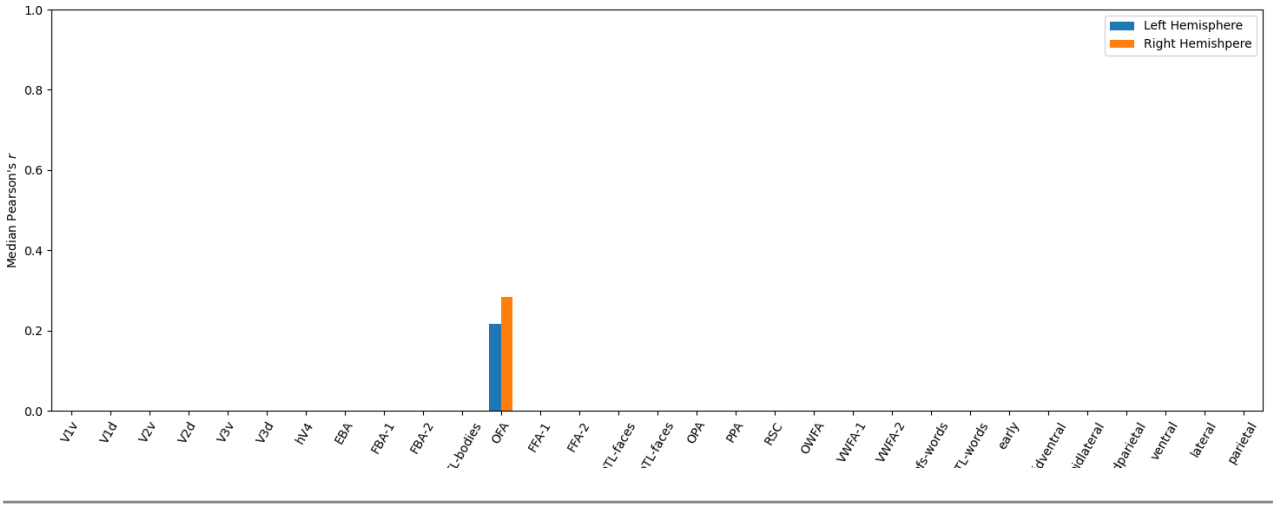


CLIP-Conv1d (Best)

Single	Yes	LR: 5e-06 Step: 10 Decay: 0.75	6 Conv1d layers, BatchNorm1d and LeakyReLU, positional attention at each step	0.3268	0.4946	0.2157
				0.3153	0.4687	0.2835

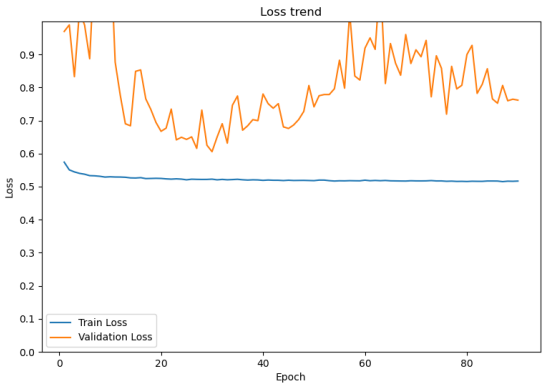


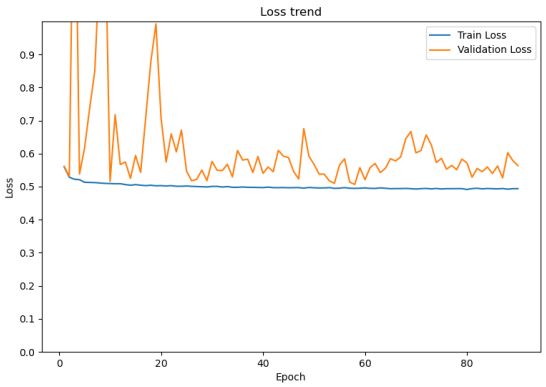




SAM-Conv1d (First)

Single	Yes	LR: 5e-06 Step: 3 Decay: 0.9	3 Conv2d layers with BatchNorm2d, LeakyReLU, Maxpool and 20% dropout. Each Conv2d layer channels were increased at each step by 1.5.	0.5152	0.6158	N/A
			6 Conv1d layers with BatchNorm1d and LeakyReLU, positional attention at each step. Each Conv1d layer channels were decreased at each step by 0.75.	0.4924	0.5067	N/A

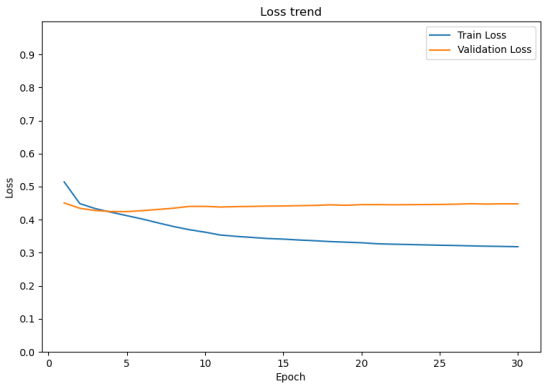


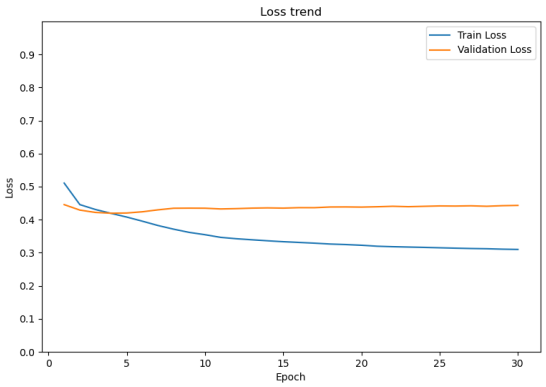


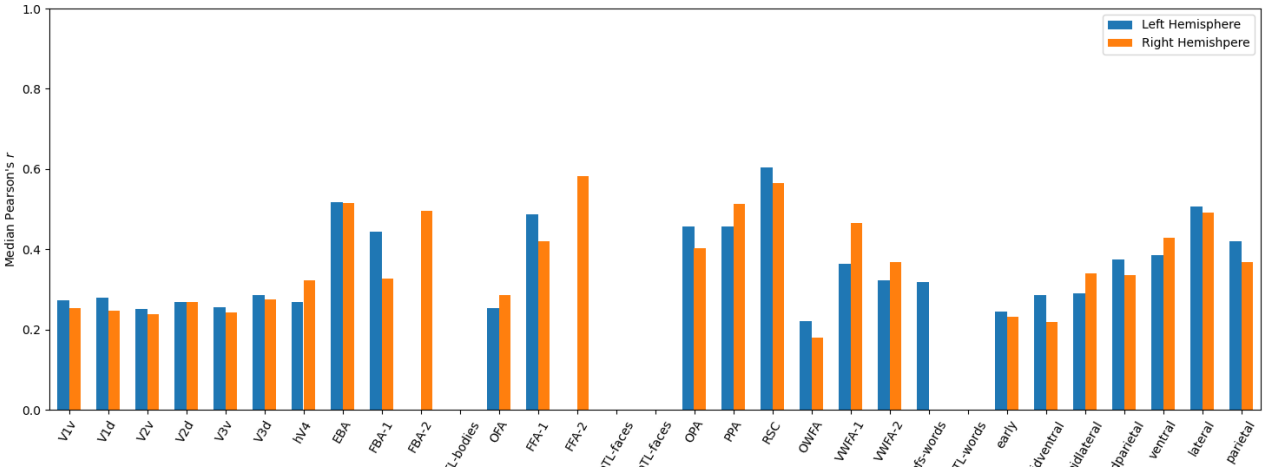
Correlation graph is not available for this experiment, but it was close to 0.

CLIP-Dropout (DNN)

All	Yes	LR: 1e-04 Step: 10 Decay: 0.5	4 linear layers with in between BatchNorm1d, LeakyReLU and 5% dropout.	0.3182	0.4245	0.2539
				0.3099	0.4195	0.2846

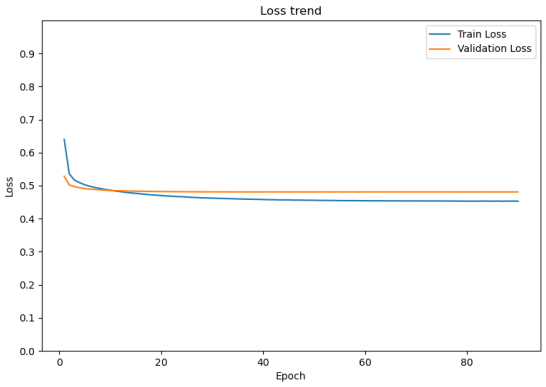


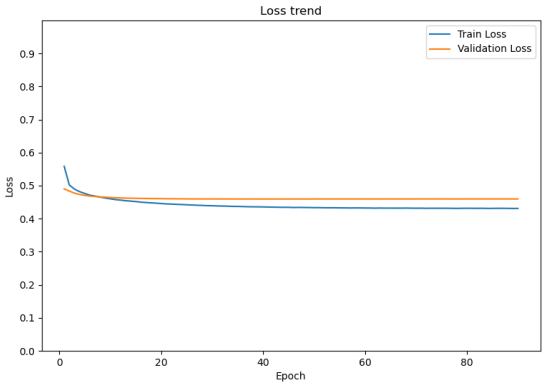


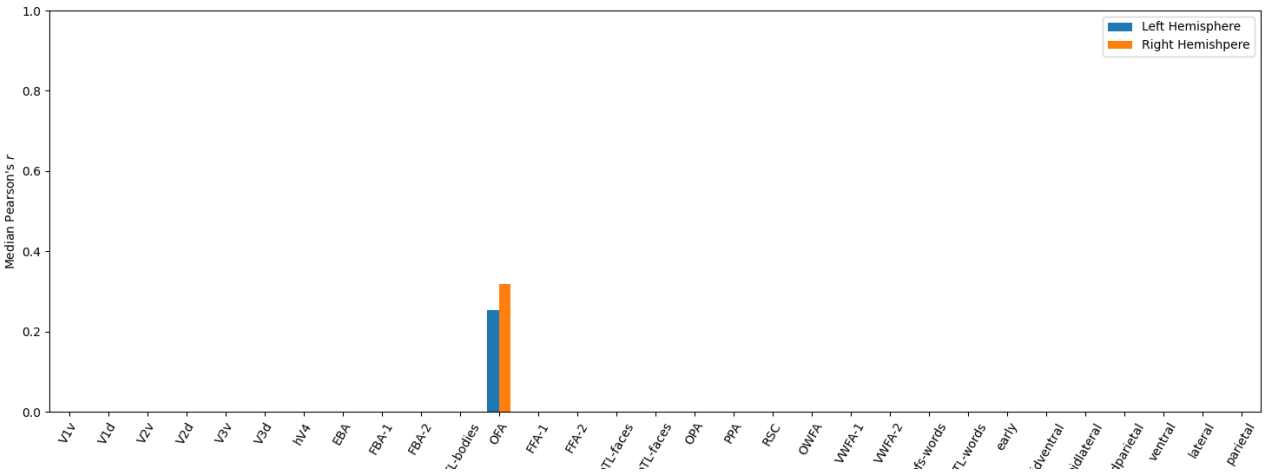


CLIP-Dropout (Conv1d)

Single	Yes	LR: 7.5e-06 Step: 5 Decay: 0.75	6 Conv1d layers, BatchNorm1d and LeakyReLU, positional attention at each step, 20% dropout every 2 steps	0.4529	0.4809	0.2528
				0.4311	0.4596	0.3180

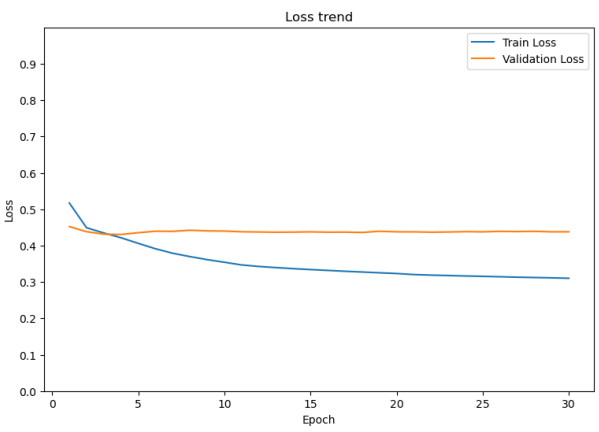


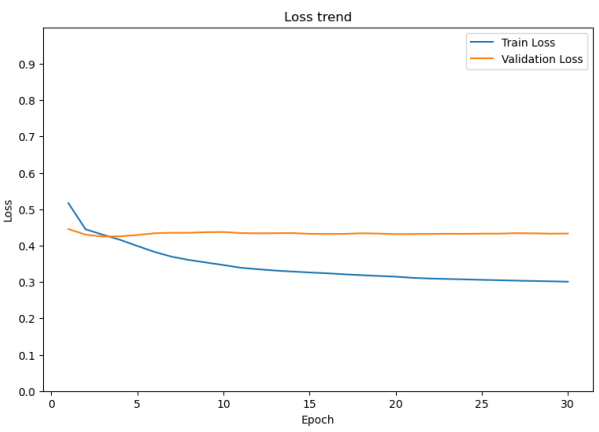


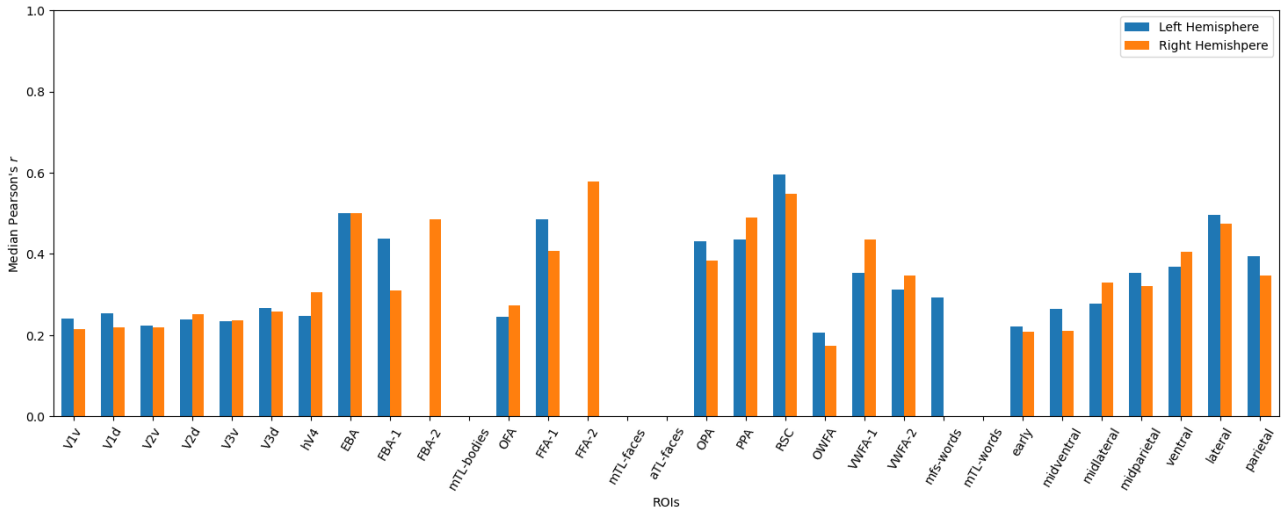


CLIP-Resblocks (DNN)

All	Yes	LR: 5e-05 Step: 10 Decay: 0.5	4 linear layers with in between BatchNorm1d, LeakyReLU and 5% dropout. After the 1st and 3rd layers, resblocks	0.3101	0.4305	0.2461
				0.3008	0.4250	0.2724

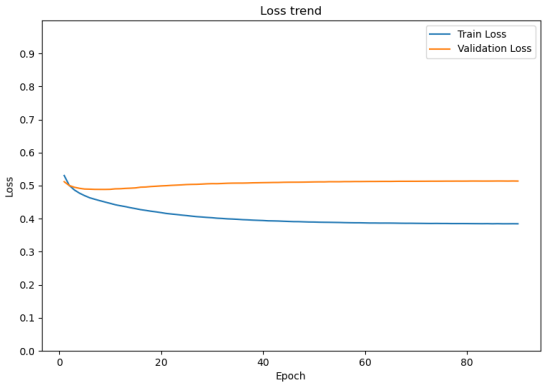


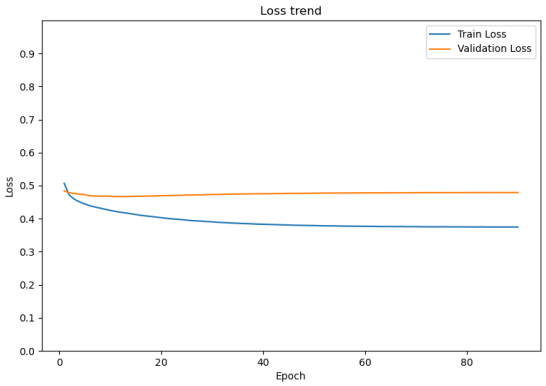


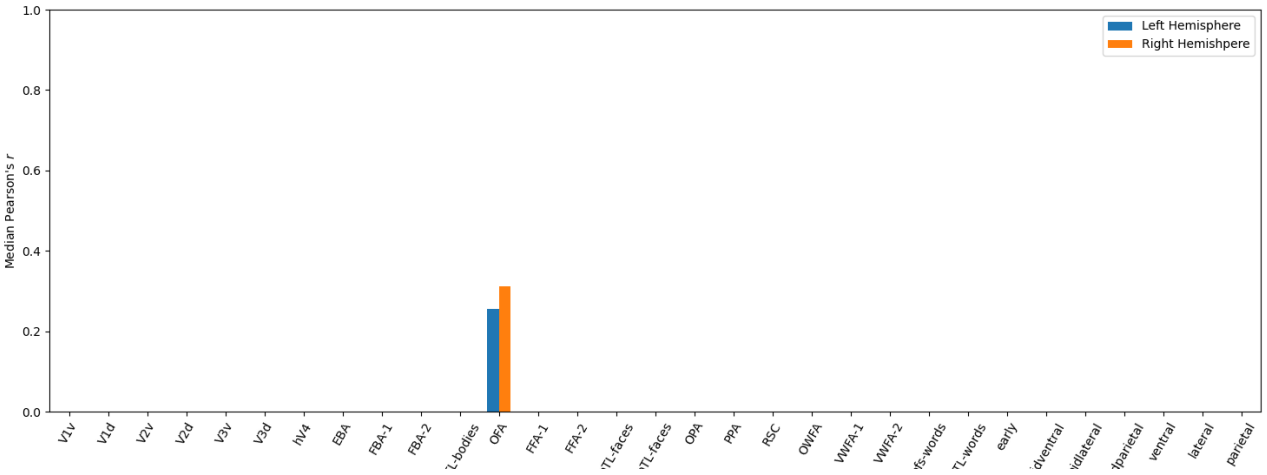


CLIP-Resblocks (Conv1d)

Single	Yes	LR: 1.5e-05 Step: 5 Decay: 0.75	6 Conv1d layers, BatchNorm1d and LeakyReLU, positional attention at each step, 20% dropout and a resblock every 2 steps	0.3846	0.4885	0.2545
				0.3747	0.4667	0.3125

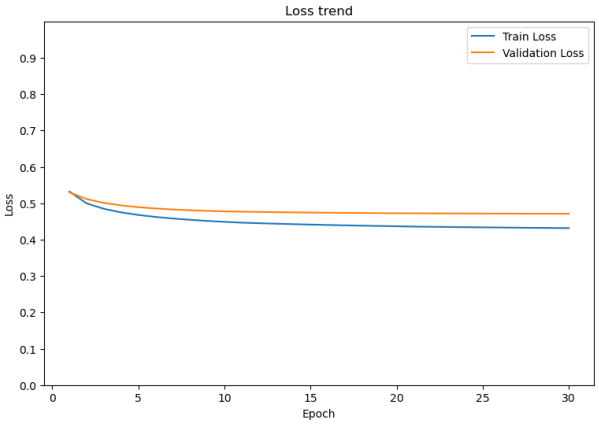


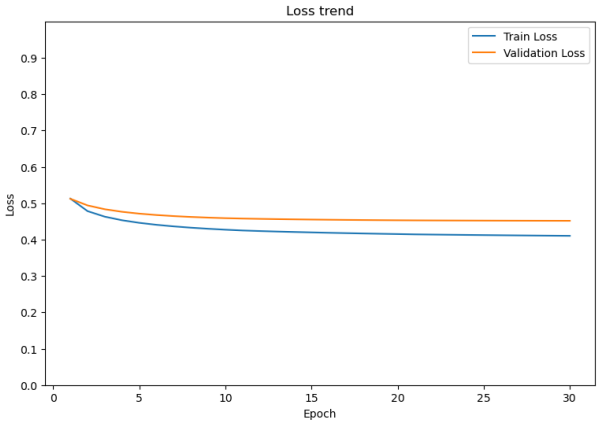


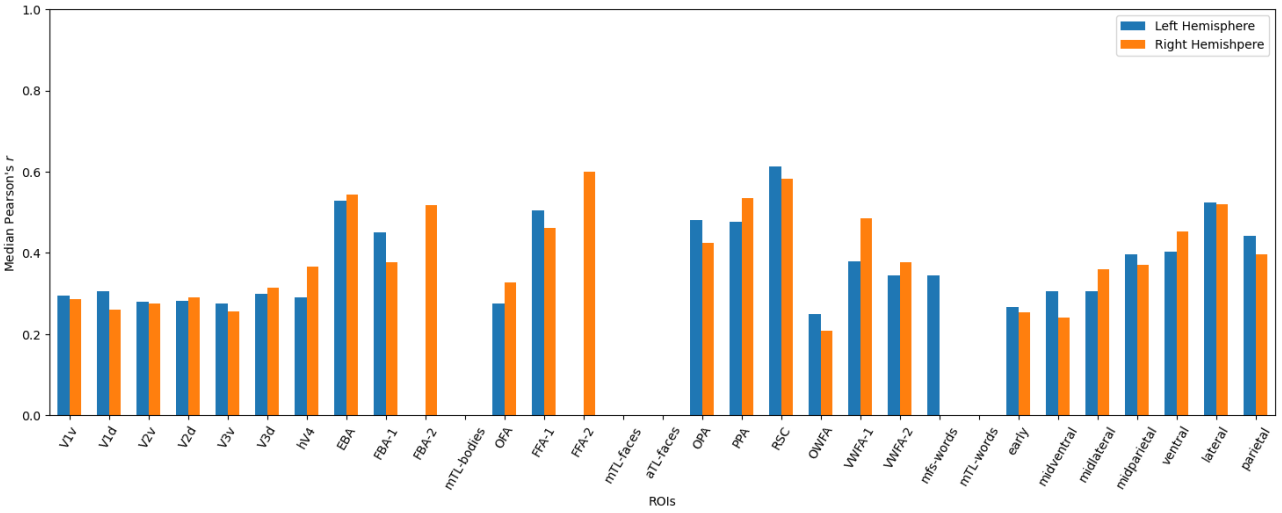


CLIP-ROI

Single	No	LR: 1e-04 Step: 10 Decay: 0.75	Single linear layer	0.4316	0.4711	0.2747
				0.4104	0.4517	0.3274

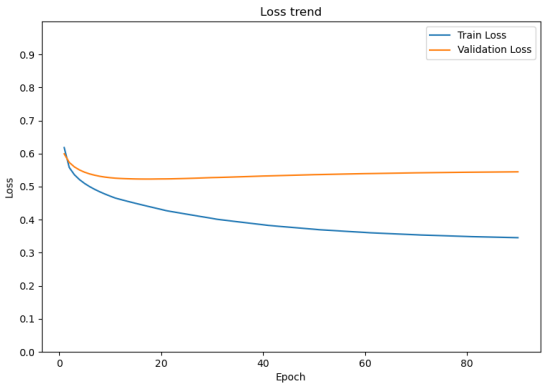


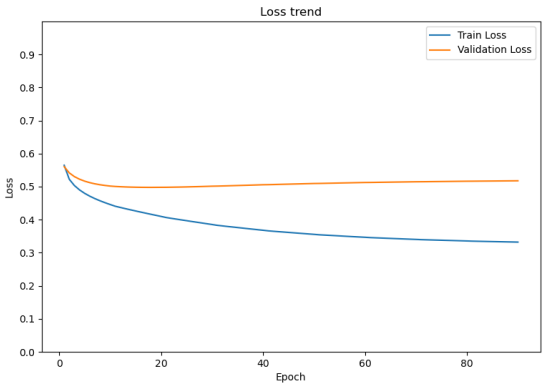


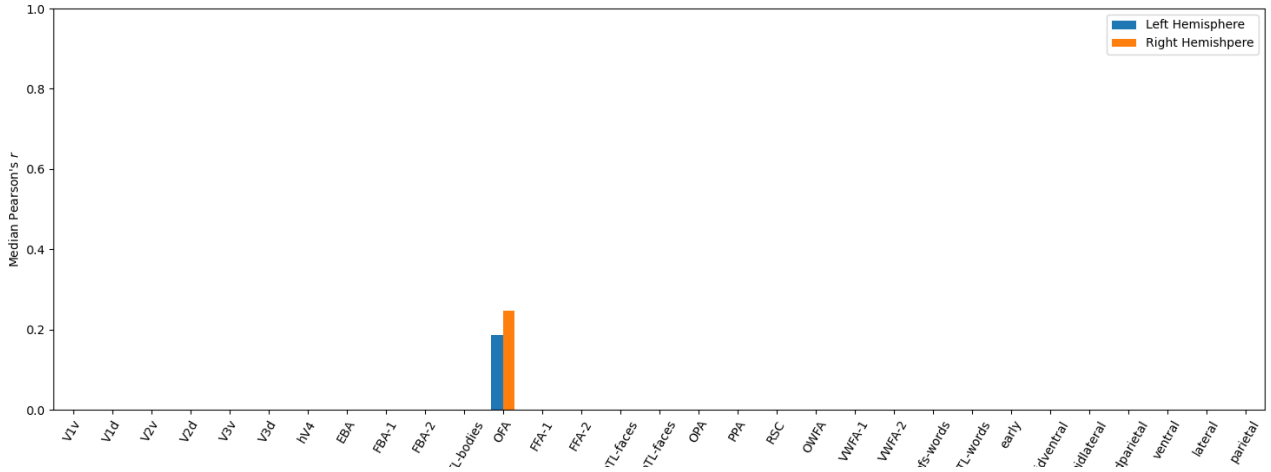


CLIP-Comb (ROI + Conv1d)

Single	Yes	LR: 7.5e-06 Step: 5 Decay: 0.75	CLIP-ROI + CLIP-Conv1d	0.3454	0.5230	0.1862
				0.3321	0.4978	0.2464

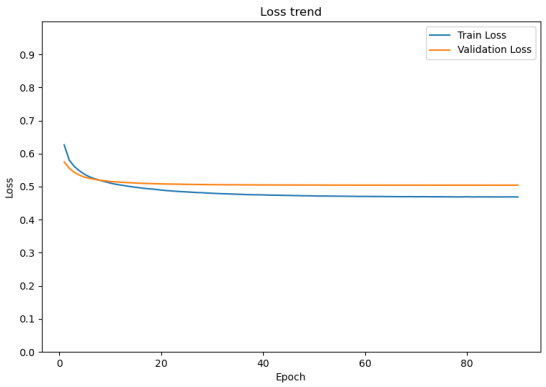


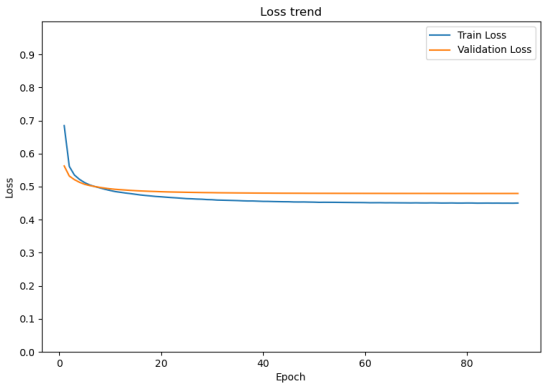


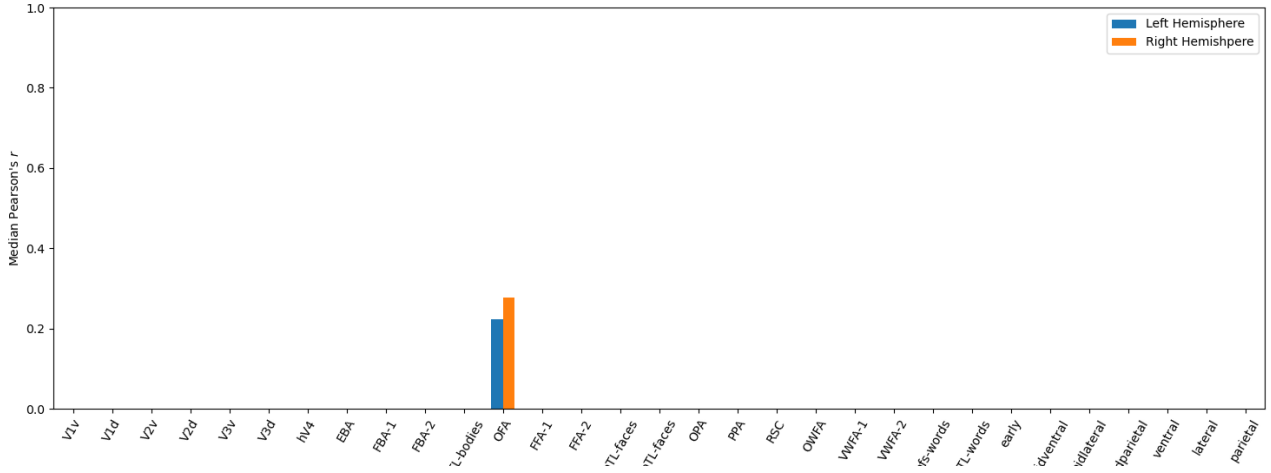


CLIP-Comb (ROI + Dropout)

Single	No	LR: 7.5e-06 Step: 5 Decay: 0.75	CLIP-ROI + CLIP-Dropout	0.4688	0.5045	0.2219
				0.4503	0.4794	0.2771

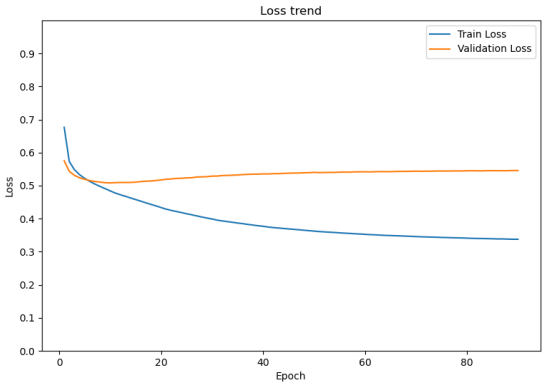


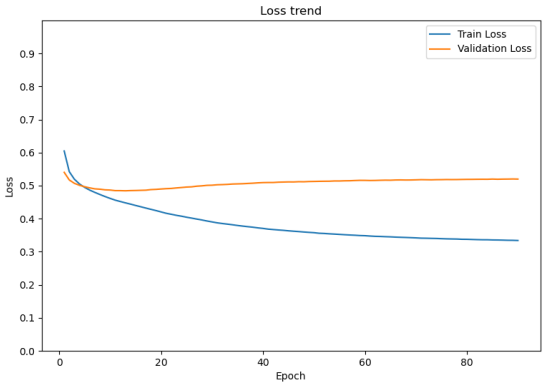


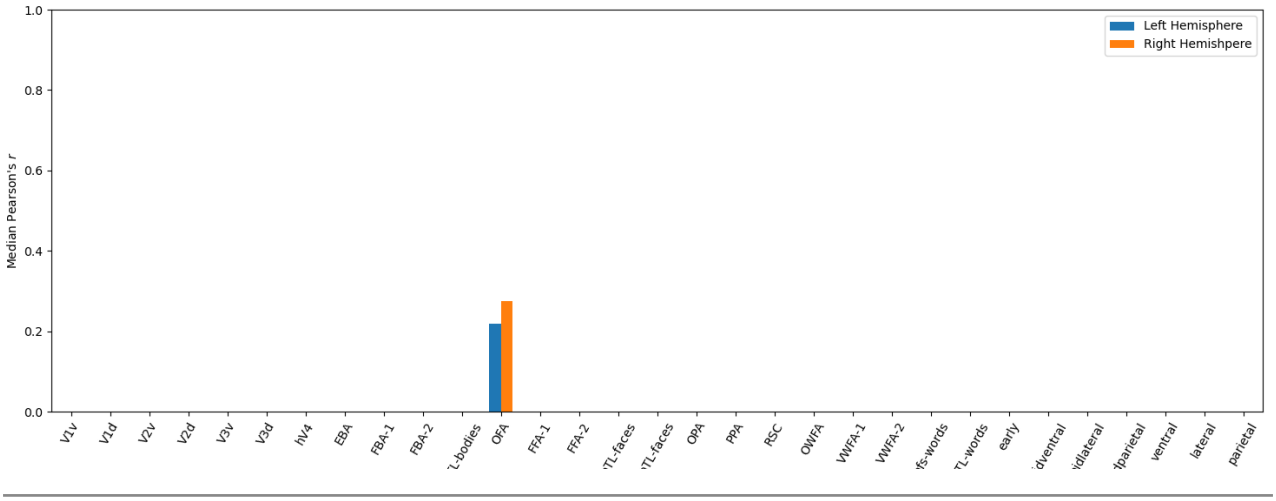


CLIP-Comb (ROI + Resblocks)

Single	Yes	LR: 1.5e-05 Step: 10 Decay: 0.75	CLIP-ROI + CLIP-Resblocks	0.3377	0.5081	0.2192
				0.3340	0.4845	0.2751







Useful links

- Up to date GitHub repository here.
- Up to date Google Sheet with experiment results here.