מכון טכנולוגי חולון
Holon Institute of Technology

Data Science School

Utilizing Transformers for Decoding Short-Term Memories of Movies from fMRI Data

Supervised by Dr. Erez Simony 2024

Tali Aharon
034791236

Karin Shauli
312277023

1. Abstract

Large Language Models (LLM) refer to a type of artificial intelligence model designed to process and generate human-like text. Transformers are prominent models in natural language processing tasks, efficiently capturing long-range dependencies in text through the use of attention mechanisms. In addition to their classic text analysis applications, transformers can also be utilized in medical data, particularly in sequential data such as that derived from fMRI scans.

By partitioning the brain's three-dimensional structure into functional parcels (also called Regions of Interest - ROI), we sample temporal sequences of blood-oxygen-level-dependent (BOLD) signals. Applying transformers to each such spatiotemporal sequence - as if it were text - allows us to analyze patterns of neural activity within each region. This approach enables us to classify the observed BOLD signals into corresponding movie stimuli based on the brain's activity during and after movie-watching. This technique is crucial for understanding how different brain regions encode and retain visual and memory-related information.

A common method to assess these functions is by conducting fMRI scans on individuals while watching movies. The use of movies has become a central tool in studying the human brain. The rich information and natural experience during movie viewing alongside fMRI scans allow us to examine the dynamics of various brain mechanisms, particularly memory mechanisms. This understanding is essential for both basic memory research and addressing memory-related issues in diseases and disorders such as Alzheimer's, autism, and more.

In this study, we utilized a transformer-based encoder model to first explore classification of neural patterns during movie-watching. Next, we focused on classification of short-term memory processes during subsequent resting-state (just after the movies end). We examined BOLD signals from 176 subjects who watched 14 movies, each followed by a 20-second rest period (with each second corresponding to one Repetition Time, or TR). The transformer model was employed to classify the fMRI responses. After extensive preprocessing and hyperparameter tuning – adjusting attention heads, dropout rates, and averaging across spatial and temporal dimensions - we developed our most effective model.

Classification accuracy was significantly higher during the end of movies compared to the subsequent resting-state periods, as expected (59% for the last 5 TRs of the movies vs. 45% for the first 5 TRs of the resting-state at the highest-scoring ROI). While decoding memory traces during the end of the resting-state period showed low but significant accuracy (since random data accuracy would be around 7%), we observed selectivity across different ROIs (20% at the highest-scoring ROI of the Visual Network vs. 8% at the lowest-scoring ROI of the Default Mode Network). Interestingly, when we shifted to network-based classification, using distributed sets of brain regions like the Visual Network or the Default Mode Network (DMN), we observed an increase in decoding accuracy (30% for the Visual Network and 13% for the DMN at the end of rest). Finally, to address the noise inherent in single-subject fMRI recordings, we generated 'super-subjects' by averaging data from subgroups of four subjects, which led to the highest classification accuracy, with 49% on the Visual Network and 45% on the DMN for the last 5 TRs of the resting-state (for comparison, the same test for single ROIs gave 27% as accuracy for the highest-scoring ROI).

Taken together, our project demonstrates a unique and successful approach to studying short-term memories of movies across subjects by using Transformers. Our results show that we can decode memory traces of movies from subsequent fMRI resting-state activity across subjects with the highest accuracy to date, and that short-term memory of naturalistic movies tends to be more distributed than localized within the human brain.

2. Introduction

2.1. Background on movie-watching in neuroscience research

Naturalistic narrative stimuli, such as movies, podcasts, and books, have emerged as essential tools in advancing neuroscience research, offering a window into the complexities of human cognition. Movies, in particular, provide a rich sensory experience that closely resembles real-life situations, engaging multiple cognitive domains simultaneously [1,2]. They serve as powerful probes for investigating the interplay between emotion and cognition, eliciting profound emotional responses [3]. Moreover, the narrative structure of movies facilitates the study of memory and attention, as viewers are required to integrate and interpret information over time as we do in daily life [4]. Consequently, movies offer researchers a unique and valuable tool to explore fundamental aspects of human brain functions.

2.2. Background on fMRI while watching movies

The use of movies in functional magnetic resonance imaging (fMRI) studies has grown notably since 2004 when Hasson and colleagues [5], demonstrated that patterns of brain activity across large areas of the cortex synchronized among individuals while watching movies. As introduced by Hasson, the Inter-Subject Correlation (ISC) analysis is a data-driven method used to assess the similarity in neural responses across individuals experiencing a common stimulus, with a particular focus on naturalistic and context-dependent audiovisual stimuli like movies. It allows researchers to explore how different brain regions synchronize in their activity patterns without relying on predefined models or assumptions about the underlying neural processes [5, 6].

2.3. Background on ISC and ISFC analysis of memory functions

Since Hasson's groundbreaking study, ISC provided insights into many human brain functions, and more specifically into our memory: Xu et al. (2005) provided early insights by examining brain activation patterns during language processing within narratives, illuminating aspects of memory encoding and comprehension of naturalistic stimuli [7]. Building on this foundation, a study by Baldassano et al. (2017) employed movie-watching paradigms to investigate how the brain encodes and retrieves complex episodic memories, shedding light on the neural mechanisms underlying memory formation and retrieval [8]. Additionally, a study by Tompary et al. (2015) explored the distinct roles of the hippocampus and medial prefrontal cortex in memory encoding and retrieval during movie tasks, deepening our understanding of memory processes in naturalistic contexts [9]. Further advancements came from Schapiro et al. (2016) using ISC to explore the statistical learning of temporal community structure in the hippocampus. [10]. While ISC reveals synchronized activation patterns within specific brain regions across individuals, ISFC, as discussed by Simony et al. (2016) [11], goes a step further. It clarifies the shared functional connectivity patterns influenced by external, temporally aligned stimuli. These studies collectively demonstrate the evolution of research in memory function using movies and fMRI, highlighting the importance of naturalistic stimuli in understanding the neural mechanisms of memory.

2.4. What is missing in the field

We need a better understanding of our short-term memory:

Despite the advancements we have just described, a gap remains in our understanding of short-term memory processes following movie-watching. What happens in the brain in the few seconds just after the movie ends? Can we shed light on short-term memory processes using ISC and ISFC? While prior studies have been essential in explaining memory formation, particularly during pauses or transitions, they primarily concentrate on episodic memory, which heavily involves the hippocampus and focuses on memories of daily events (such as times, locations, associated emotions, and contextual details). These studies do not definitively disclose the nature of short-term memory traces - Do they only exist in specialized memory-related networks, or are they part of a broader cortical mechanism that can be categorized based on different timescales [12].

2.5. Our Research description - based on the Transformer architecture

To address these gaps and try to bridge between short-term memory and episodic memory, here we wish to explore brainwide short-term memory traces using naturalistic stimuli. This project aims to classify fMRI responses of rest periods between movies into 14 different movies. We aim to utilize large-language models (LLMs) to study neural translation from movie-induced neural representations to memory traces [13]. The encoder-decoder Transformer architecture, a key building block of LLMs, will be employed for this purpose [14]. This capability is demonstrated by the recognition that transformers, traditionally employed in language processing, extend their applicability to the processing of extensive sequences. Specifically, they exhibit proficiency in handling diverse data types, including time series data captured by fMRI [15]. Recent advances in machine learning, particularly in the field of representation learning (a process where algorithms automatically discover and utilize features of data necessary for tasks like classification or prediction, which enables algorithms to construct representations of the data that are more understandable to further processing [16]), have shown promise in enhancing our understanding of neural processes. For instance, Zerveas et al. (2021) presented a Transformer-based framework for multivariate time series representation learning, demonstrating the potential of Transformer models in capturing complex temporal relationships in data [17]. Additionally, Malkiel et al. (2022) used Self-Supervised Transformers for fMRI representation learning, showing that a pre-trained model which is fine-tuned on specific tasks yields promising results in capturing neural representations from fMRI data [18]. These advancements suggest that Transformer models could be valuable tools in decoding short-term memories of movies from fMRI data, offering new insights into human cognition.

2.6. Our Dataset and Methods

Our dataset comprises time-series of the blood-oxygen-level-dependent (BOLD) signal in fMRI, taken from the Human Connectome Project, where 176 subjects watched 14 movie clips interlaced with resting-state intervals. We preprocessed this data to prepare it for analysis. First, we trained a Transformer model to classify fMRI responses to movie names, establishing a baseline for understanding the neural correlates of movie recognition. Next, we applied the same model to classify fMRI responses during resting-state intervals after each movie, identifying neural patterns associated with short-term movie memory. Our focus was on intervals of approximately 15-20 seconds after the movie clip ends, a time frame sufficient for hemodynamic decay [19].

2.7. What this research will contribute to the field

Our research contributes to the field by offering deeper insights into short-term memory processes during and after movie-watching, using fMRI data and advanced machine learning techniques. By applying large-language models and the Transformer to neural representations of movie clips and memory traces, we demonstrated the ability to accurately classify which movie an individual watched based on just 5 seconds of memory traces. Our findings also highlight that memory processes are best understood within the framework of large-scale brain networks. This research holds potential implications for understanding memory disorders and informing the development of therapeutic interventions. Ultimately, our study aims to bridge the gap between naturalistic stimuli, neural activity, and memory processes, thereby advancing our understanding of human cognition.

3. The Brain - Background

3.1. The Essential Role of the Human Brain in Body Functionality

The human brain serves as a fundamental organ within the body, playing a crucial role in the nervous system to facilitate the operation of bodily systems. Its primary function involves processing information received from sensory systems and generating appropriate responses based on that information. Comprising three key structures, the human brain includes the Cerebrum, which dominates the cranial cavity and is divided into the right and left hemispheres connected by the Corpus Callosum; the Brainstem, situated in the lower part of the brain and linked to the spinal cord; and the Cerebellum, positioned behind the Brainstem and beneath the Cerebrum. Functionally, the Brainstem regulates basic bodily functions such as breathing and arousal, acting as a conduit for sensory information and motor commands between the brain and spinal cord. The Cerebellum primarily coordinates precise motor actions. The processing of sensory information and the execution of voluntary body movements are governed by the Cerebrum, encompassing its diverse components. Furthermore, the brain's tissue can be broadly categorized into gray matter, containing nerve cell bodies, and white matter, primarily composed of axons that interconnect various regions of gray matter. In addition to neuronal tissue, the brain houses a system of cavities known as ventricles, filled with cerebrospinal fluid (CSF), distributed between the brain's layers and along the spinal cord pathway. The Cerebrum is further subdivided into four lobes: the Frontal lobe, the Temporal lobe, the Occipital lobe, and the Parietal lobe.

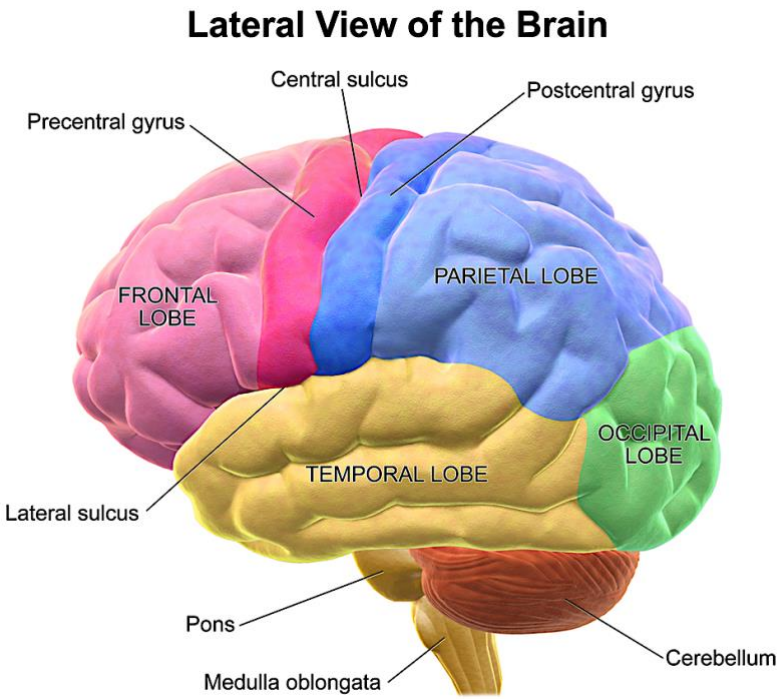

Figure 1 - Four lobes of the brain [20]

3.2. fMRI - Functional Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is a type of non-invasive scan used to visualize internal organs in the body without exposure to ionizing radiation. This method combines RF radiation and a strong magnetic field to influence the spin direction of protons in the hydrogen atomic nucleus found in body water molecules. During the scan, the subject is placed in a constant magnetic field that aligns the spin direction of all protons. By sending radio wave pulses, a precession motion of proton spin (phase alignment and spin direction deviation) is created. This causes the proton to generate a resonance signal that can be measured using MRI machine coils, allowing clear differentiation between different tissues. The spin properties change depending on the tissue it is in (e.g., white matter tissue or gray matter tissue in the brain).

Functional Magnetic Resonance Imaging (fMRI) is a method of magnetic resonance imaging that measures brain activity. Unlike traditional MRI, fMRI does not show a static image of tissue structure but captures multiple images at a high frequency reflecting the functional aspects of tissues. This method is based on the magnetic properties of hemoglobin protein, which carries oxygen and carbon dioxide molecules. Oxygenated hemoglobin reacts differently from deoxygenated hemoglobin, allowing differentiation between various states of hemoglobin and determining if a specific brain region requires more oxygen. This measurement is known as Blood-Oxygen-Level Dependent (BOLD). The results of fMRI scans do not directly show electrical activity but indirectly through BOLD, measured at a resolution of approximately 1-2 seconds.

3.3. Important parameters of fMRI scans

- **Volume:** The Volume (3D Image) of the brain made of slices.
- **Slice:** Dividing the brain into horizontal slices that compose a three-dimensional voxel image of the brain. These slices are scanned sequentially by the scanner.
- **Voxel:** Dividing each slice into individual units called voxels, which are essentially pixels in three-dimensional space. Each fMRI image contains approximately 100,000 voxels.
- **Repetition Time (TR):** The time between RF pulses. A long TR allows protons in tissues to realign with the main magnetic field. A short TR causes some tissue protons not to realign with the magnetic field before the next measurement, reducing the signal from that tissue.

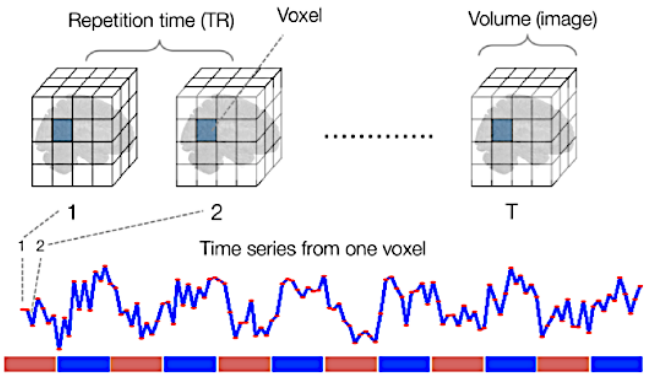

Figure 2 - fMRI Data Time Series [21]

3.4. Brain Networks

In recent years, there have been numerous attempts to investigate the various functional brain networks. These networks consist of multiple brain regions that establish connections with each other at specific times to perform a particular function. These functional networks are identified, among other methods, through fMRI scans.

We can divide the human brain into 7 functional networks, each associated with distinct cognitive and sensory functions:

- **Visual Network (Vis):** Encompasses regions primarily involved in visual processing, including the primary visual cortex and other visual areas. It is essential for processing visual information from the environment, such as shape, color, and motion.
- **Somatomotor Network (SM):** Includes regions that control voluntary movements and process sensory information. It encompasses the primary motor cortex and somatosensory cortex, facilitating activities such as movement execution and tactile sensation.
- **Dorsal Attention Network (DAN):** Responsible for goal-directed attention and spatial awareness. It includes regions like the frontal eye fields and intraparietal sulcus, playing a critical role in tasks requiring focused attention and visual tracking.
- **Ventral Attention Network (VAN):** the VAN, or salience network, is involved in detecting and orienting to salient stimuli. Key regions include the anterior insula and the anterior cingulate cortex, which help prioritize sensory information and guide adaptive behavior.
- **Limbic Network (Limbic):** Comprises areas related to emotion, memory, and motivation, such as the orbitofrontal cortex and temporal pole. It is crucial for processing emotional stimuli and forming memories, linking sensory experiences with emotional responses.
- **Frontoparietal Network (FPN):** The FPN, or executive control network, is engaged in high-level cognitive functions like working memory, decision-making, and cognitive flexibility. It includes regions such as the dorsolateral prefrontal cortex and posterior parietal cortex, facilitating goal-oriented behavior and problem-solving.
- **Default Mode Network (DMN):** Active during rest and self-referential thought, including daydreaming, recalling memories, and envisioning the future. It involves areas like the medial prefrontal cortex, posterior cingulate cortex, and inferior parietal lobule, supporting introspection and social cognition.

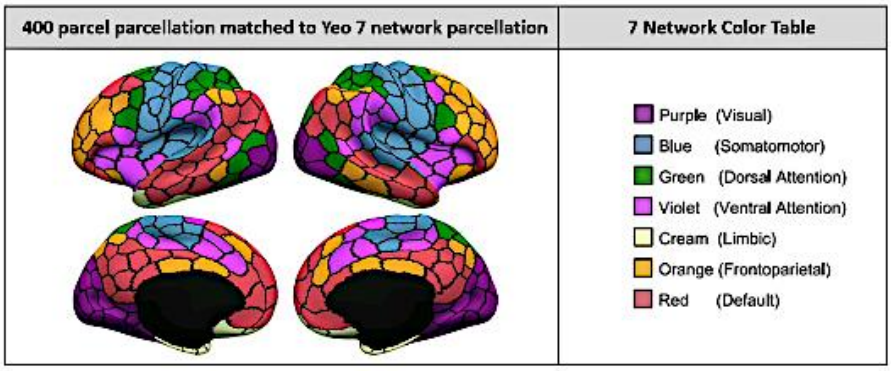

Figure 3 - Coordinate data for the Schaefer et al. (2018) cortical atlas, with 400 regions on the cortical surface of the brain, aligned with 7 previously described functional networks (Yeo et al., 2011). [22]

4. Deep Learning

4.1. Machine Learning - Background

Machine learning is a subfield of computer science and artificial intelligence that has evolved into a multidisciplinary domain. It was defined by Arthur Samuel as "the field of study that gives computers the ability to learn without being explicitly programmed." Through machine learning algorithms, computers can be trained to analyze results by performing an analysis of various datasets, learning how to use them for tasks such as prediction and identifying recurring patterns.
Several key terms are essential in this field:

- **Observation:** In machine learning, an observation refers to a single instance or data point within a dataset that contains information about the variables being studied.
- **Feature:** Individual measurable properties or characteristics of an observation that are used as input variables in machine learning algorithms to make predictions or classifications.
- **Classification:** Classification is a type of machine learning task where the goal is to categorize or assign a label to a given observation based on its features, typically into predefined classes or categories.
- **Regression:** Regression is a machine learning technique used to predict continuous numerical values based on the relationship between input features and output variables, aiming to find a function that best fits the data points.
- **Dataset:** A sequence of labeled/classified examples, where the correct classification of the input pattern is known. In this field, it is important to have a wide range of examples to enable the construction of a classifier that will classify any new input into the appropriate class with minimal error. The Dataset is usually split into Training, Validation and Test sets (in a ratio of 70%-15%-15% or other).
- **Training Set:** The training process in machine learning involves iteratively adjusting the model's parameters using an optimization algorithm on a labeled Training Set to minimize the error between the predicted outputs and the actual targets, aiming to improve the model's performance and ability to make accurate predictions on unseen data.
- **Validation Set:** Used for model selection and tuning.
- **Test Set:** Used for final performance evaluation.
- **Loss Function:** Quantifies the difference between the model's predictions and the actual target values. It guides the optimization algorithm in updating the model parameters. Common loss functions include mean squared error (MSE) for regression tasks and cross-entropy loss for classification tasks.
- **Bias-Variance Tradeoff:** Refers to the delicate balance between the model's ability to capture the underlying patterns in the data (bias) and its sensitivity to fluctuations or noise in the data (variance), highlighting the need to find a suitable model complexity that minimizes both sources of error for optimal predictive performance.
- **Overfitting and Underfitting:** Overfitting occurs when a machine learning model learns the noise and random fluctuations in the training data to the extent that it performs well on the training data but fails to generalize to new, unseen data, while underfitting happens when the model is too simple to capture the underlying patterns in the data, resulting in poor performance on both the training and test datasets.

4.2. Deep Learning and Artificial Neural Networks Structure

Deep learning is a subfield of machine learning that utilizes artificial neural networks to perform tasks. At its core, this field aspires to mimic the way the human brain operates and harness the efficiency of neural structure to tackle complex computational challenges.

When considering a Neural Network in comparison to the human brain, the Perceptron can be likened to the individual neurons. A Perceptron functions as a singular unit with a specific role, and multiple Perceptrons collaborate to construct a Neural Network. This processing unit is named due to its resemblance to a physiological neuron - the basic processing unit in the human brain.

The input to the perceptron is a set of inputs (x_1 to x_n) multiplied by weights (w_1 to w_n) and a bias element (b). The input goes through a Summation (the linear function, z), then the sum passes through an activation function f and yields the Output y.

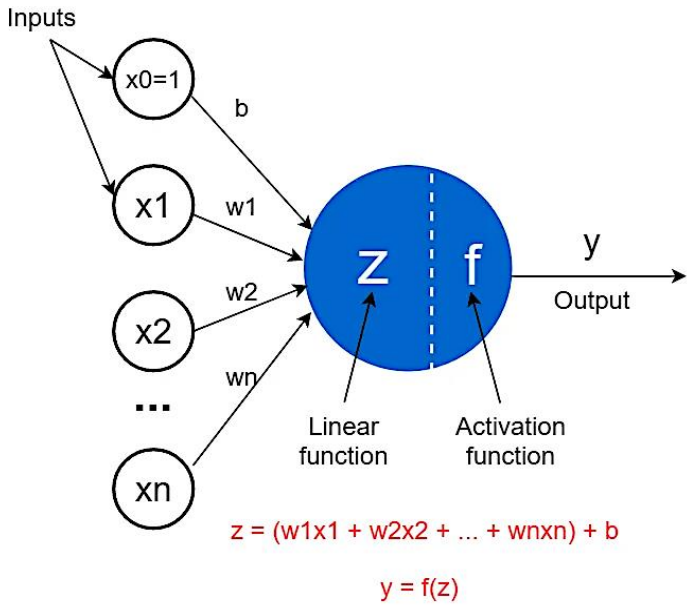

Figure 4 - The structure of a perceptron [23]

In practice, this involves a computational mathematical model consisting of a number of "neurons" arranged in layers, where each neuron can connect with other neurons in the system. Each neuron can perform simple computational operations and transmit the derived information to other neurons. As information progresses through the layers, the system transforms raw data into valuable information. Consequently, the system learns to make more precise decisions.

In cases where the neurons connected to the input do not constitute the output but are multiplied by weights and connected to another layer of neurons, the connected layer to the input is called a hidden layer. If there is more than one hidden layer, the network is called a deep neural network.

The connection between the layers is done in the same way - weight multiplication, summation, and passing through the activation function.

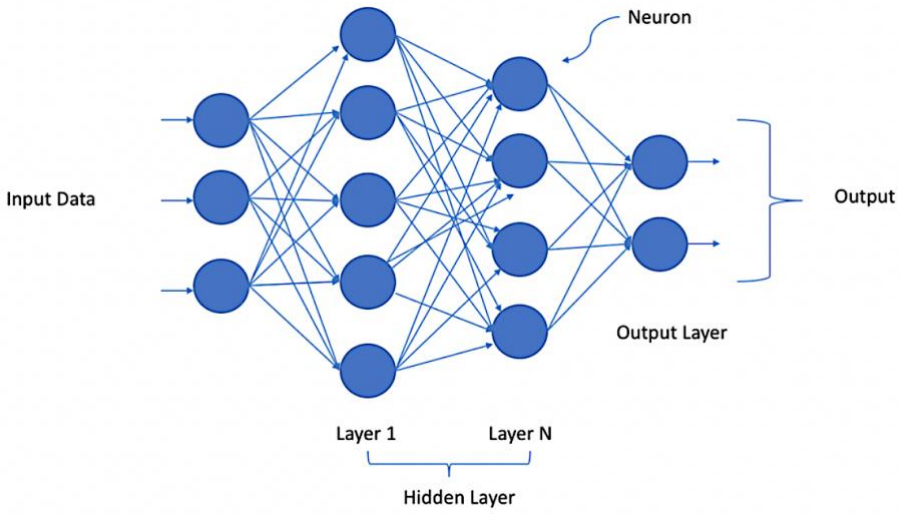

Figure 5 - A Deep Neural Network with N hidden layers [24]

4.3. Activation Functions

Activation functions introduce non-linearity into the network, enabling it to learn complex patterns. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh. This helps the network to approximate more complex functions and improve its learning capacity.

4.4. Training an Artificial Neural Network

Training a neural network involves the iterative process of adjusting the weights and biases of the network to minimize the error between the predicted outputs and the actual targets. This error is quantified using a loss function, such as Mean Squared Error (MSE) or Cross Entropy, which measures the difference between the predicted and actual values.

During forward propagation, input data is passed through the network to make predictions, and the error is calculated using the chosen loss function.

In the subsequent phase known as backward propagation or backpropagation, the gradients of the loss function with respect to the weights and biases are computed. Backpropagation is a technique that calculates these gradients by propagating the error backward through the network. These gradients indicate the direction and magnitude of adjustments needed to minimize the error, and the weights and biases are updated accordingly using an optimization algorithm like gradient descent.

This iterative process of forward and backward propagation, guided by the loss function and gradients, helps the neural network learn from the data and improve its predictive performance over time.

4.5. Transformers and the Attention Mechanism

In the context of neural networks, the application of the Attention mechanism is characterized by dynamically emphasizing the most relevant information from the given data and utilizing prominent parts of it. In the Attention network, the input sequence is initially processed by an encoder, which, through the Attention mechanism, extracts the important features in the data that the decoder can use to generate the output by assigning weights and creating a context vector representing the relationships between all data components. The role of the decoder is to generate the output according to the specified task (such as classification, regression, etc.). In "Attention" networks, both the encoder and the decoder make use of an Attention mechanism consisting of processing units called heads to identify patterns and relationships between data sequences.

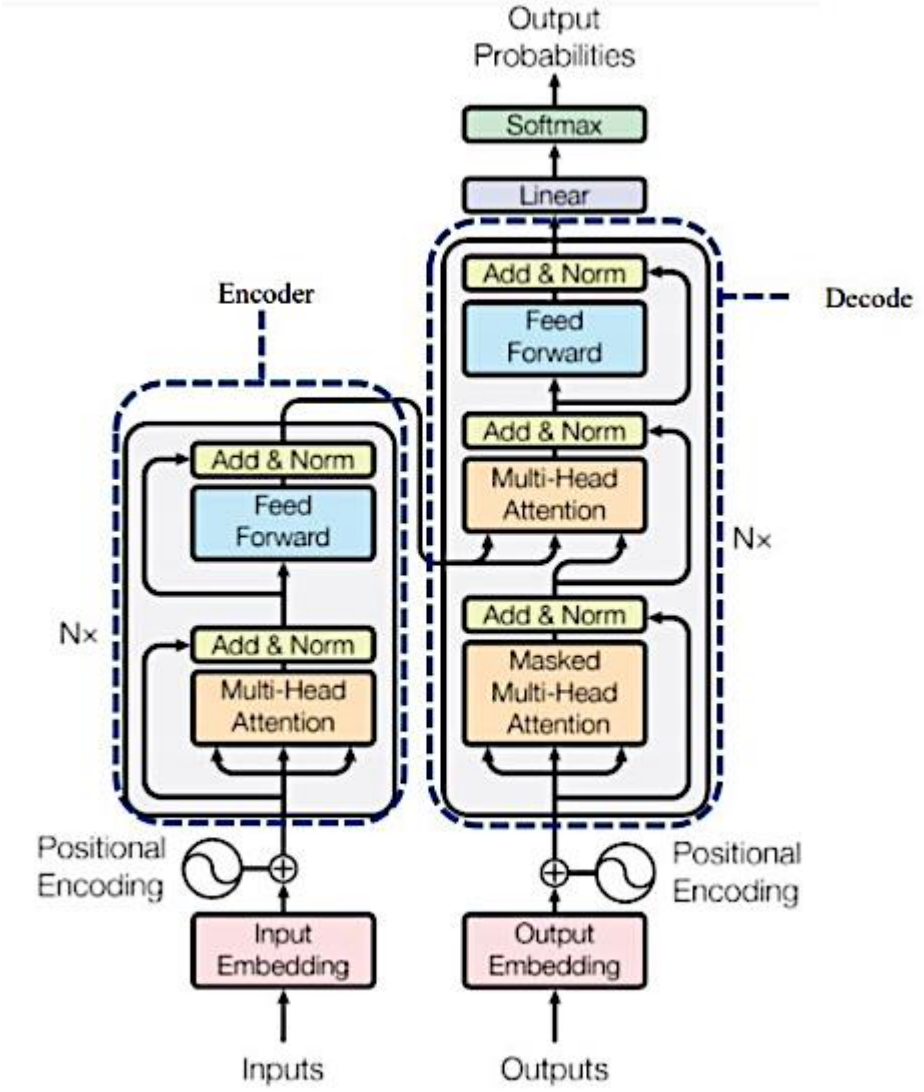

Figure 6 - The transformer Structure [14]

Attention networks can be used in their full form (encoder and decoder) for tasks like language translation and regression problems. However, it is also possible to utilize only the encoder for tasks such as classification and prediction (e.g., predicting the next word in a sentence).

The commonly used layer types used in the transformer structure are:

- **Linear Layer:** Transforms input features into a different space, preparing them for the attention mechanism.
- **Multihead Attention Layer:** Computes attention scores by focusing on different parts of the input simultaneously, enhancing the model's ability to capture complex patterns.
- **Dropout Layer:** Randomly zeroes some elements of the input tensor during training to prevent overfitting and improve generalization.
- **LayerNorm:** Normalizes the input across the features, ensuring stable training and faster convergence.
- **Position-wise Feedforward Layer:** Applies a linear transformation followed by a non-linear activation function to each position independently, enabling complex mappings.
- **Positional Encoding:** Adds information about the position of each element in the sequence, allowing the model to understand the order of the inputs.

The Attention mechanism generates a context vector that represents the essence of the input. The context vector is composed of Query, Keys, and Values vectors created by representing the input in three different ways using learned weight matrices. Queries represent individual elements in the input sequence and serve for extracting information from the Embedding vector. Keys represent the memory component and are used to match the Query vector to calculate the weight for each element in the input sequence. Values represent the associated information to the Key and are weighted accordingly to generate the context vector, which is the output of the Attention mechanism.

To calculate the Attention weights for each input element, the multiplication between the Query and Key is computed. The weights are normalized using the Softmax function and are used to calculate the weight of the corresponding Value vector. Finally, with these three vectors, the context vector Z is computed, as depicted in Figure 7.

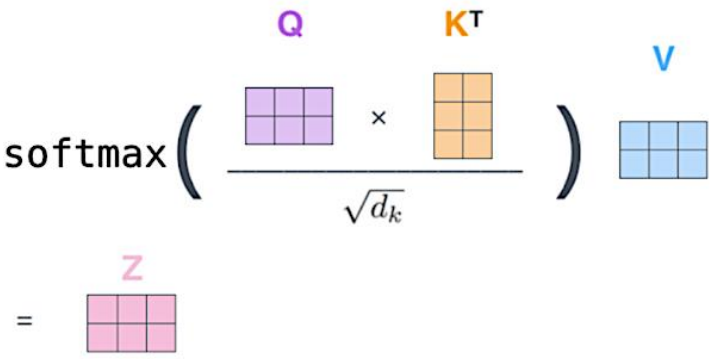

Figure 7 - Calculating the context vector in the attention mechanism [25]

5. Model Evaluation

Model evaluation in deep learning shares similarities with traditional machine learning evaluation techniques but introduces specific considerations due to the complexity of deep neural networks. Here is a structured approach to model evaluation in deep learning:

5.1. Train-Validation-Test Split

As in traditional machine learning, the data is generally divided into three sets: a training set for model training, a validation set for hyperparameter tuning and model selection, and a test set for the final evaluation of the selected model.

5.2. K-Fold Cross Validation

Cross-validation is often used to ensure robust model evaluation in cases where the dataset is small. This technique involves partitioning the dataset into K subsets (folds) of equal size. The model is trained on $K-1$ folds and validated on the remaining fold, with this process repeated K times so that each fold serves as the validation set once. The performance results from each iteration are averaged to provide a more reliable test accuracy.

5.3. Metrics for Classification

- **Loss:** The measure of the difference between the predicted values and the actual values in a dataset, used to quantify the performance of a model.
- **Accuracy:** The ratio of correctly classified instances to the total number of instances.

5.4. Early Stopping

To mitigate overfitting, early stopping is often employed. Training is halted when performance on the validation set begins to deteriorate, as indicated by a monitored metric such as validation loss.

5.5. Visualizations

Visualization techniques, including tables of accuracy results, learning curves, and heat maps, provide additional insights into model performance and behavior, facilitating a deeper understanding of the model's strengths and weaknesses.

6. Dataset

The Human Connectome Project (HCP) was initiated by the National Institutes of Health (NIH) in July 2009 as one of the three major challenges in the NIH's brain research program. The primary goal of the project is to build a database of "brain maps" that organize the human brain's structural and functional connectivity, aiming to create a resource for studying various cognitive functions of the human brain based on brain mapping.

Within the project, a dataset of accessible information is being constructed, containing resting-state fMRI scans and movie-watching fMRI scans at 3T resolution for approximately 1200 participants and at 7T resolution for around 180 participants (from the same pool of 1200 participants). Additionally, personal information about the participants (such as age, gender, ethnicity, family relationships) and results of various cognitive tests conducted on the participants are included. Over the years, several versions of data have been released. In this project, we are working with the HCP S1200 dataset released in 2016, which has undergone extensive preprocessing.

The fMRI scans at 7T were conducted in four sessions, each lasting about an hour and a quarter. Each session started with a resting-state scan (rfMRI), followed by two movie-watching scans and two scans in other conditions not relevant to this project. The rfMRI data were acquired in four runs of approximately 16 minutes each, with each run starting in a dark room with the participant's eyes open and focusing on a bright cross image on a dark background projected on the wall.

In the first and fourth runs, after the resting-state scan, two movie-watching scans were performed. Each movie-watching scan presented movies totaling about 15 minutes, consisting of four unrelated and non-sequential video clips. In total, four movie-watching scans were conducted.

All fMRI scans at 7T were performed with a repetition time (TR) of one second and a measurement time of 22.2ms, with a slice thickness of 1.6mm and 85 slices.

Parameter	Value
Sequence	Gradient-echo EPI
TR	1000 ms
TE	22.2 ms
flip angle	45 deg
FOV	208 x 208 mm (RO x PE)
Matrix	130 x 130 (RO x PE)
Slice thickness	1.6 mm; 85 slices; 1.6 mm isotropic voxels
Multiband factor	5
Image Acceleration factor (iPAT)	2
Partial Fourier (pF) sampling	7/8
Echo spacing	0.64 ms
BW	1924 Hz/Px

Condition	Runs	Frames (TRs) per run	Run Duration (min:sec)
REST (Resting-state)	4	900	16:00

Figure 8 - The HCP full scan protocol [26]

7. Methods

7.1. Motivation and Overview

Large Language Models (LLMs), such as transformers, excel in processing and generating human-like text by capturing long-range dependencies through attention mechanisms. Beyond text, transformers can analyze sequential medical data like fMRI scans. By dividing the brain into functional regions (parcels) and sampling temporal sequences of blood-oxygen-level-dependent (BOLD) signals, transformers can predict BOLD signals from prior activity and infer inter-regional dependencies. This approach is pivotal for understanding brain activities such as motor, language, visual, and memory processes.

This project employs a unique dataset of individuals watching 14 different movies with rest periods in between to classify fMRI responses to movies and memory representations using transformers. By leveraging the capabilities of transformer models, we investigate the fMRI scans taken while watching movies and compare them to the ones taken in the rest between movies. The project aims to understand the relations between neural mechanisms underlying short-term memory (rest) and the neural representation of movie-watching experiences. The integration of fMRI data with advanced machine learning techniques offers a novel approach to understanding human cognition.

7.2. The fMRI Preprocessing

The preprocessing pipeline ensures that the fMRI data is properly formatted, normalized, and segmented, making it ready for training and evaluation with the transformer model. The dataset was divided into three sets: 80% for training, 10% for validation, and 10% for testing.

Key Steps in Our Preprocessing:

1. **Initialize Dataloaders:** Create a dictionary to store dataloaders for each phase (train, eval, test).
2. **Iterate Over Phases:** Loop through each phase to load the corresponding dataset.
3. **Process Each Subject's Data:** For each subject folder, check if the required file exists. If the file exists, load the data and determine the number of voxels by subtracting the last three columns (assumed to be labels and other metadata).
4. **Slice Data:** Depending on the specified slice (start, end, middle, all), select a portion of the fMRI TRs sequence for each movie/rest segment (portion length controllable by parameter). For the "all" slice, the data was first padded with zeros, but this introduced bias, so we decided to use the shortest movie length and only consider this number of TRs for all movies (65 TRs).
5. **Prepare Outputs:** Create one-hot encoded labels for each movie.
6. **Create Tensor Datasets and Dataloaders:** Stack the inputs and outputs into tensors. Then create PyTorch TensorDataset and DataLoader objects for each phase.
7. **Return Dataloaders:** Return the dataloaders for train, val, and test phases, along with the number of voxels.

7.3. Training the Classification Model - Overall Structure

The training code ensures a systematic approach to training, validating, and monitoring a transformer-based model for fMRI data.

The Training Process:

1. **Configuration and Initialization**: Set hyperparameters, initialize Weights & Biases (wandb) package, and determine the device.
2. **Data Loading**: Load and preprocess the fMRI data.
3. **Model Initialization**: Create the transformer model, optimizer, and loss function.
4. **Training Loop**: Train the model over multiple epochs, compute and log training loss and accuracy.
5. **Validation Loop**: Evaluate the model on validation data, compute and log validation loss and accuracy.
6. **Logging**: Log metrics using wandb.
7. **Model Saving**: Save the model state periodically.

7.4. The Model - Classification using Transformer Encoder

Each layer in the model is designed to transform the fMRI data, capture complex patterns, and ultimately make accurate predictions on the classification tasks. For the classification tasks, we used only the Transformer Encoder, without the decoder.

Layers in the Transformer Model:

1. **Input Linear Layer:** Converts the input voxel data to a higher-dimensional space. This transformation is crucial for enabling the model to capture complex patterns in the data.
2. **Positional Encoding:** Adds positional information to the input data to capture sequence order. This step ensures the model can understand the temporal order of the fMRI signals, which is essential for time-series data.
3. **Attention Layers:** Multiple layers of attention blocks capture dependencies within the data. Each attention block consists of a multi-head attention layer followed by a position-wise feedforward network. Layer normalization and dropout are applied after each sub-layer. The attention mechanism helps the model focus on different parts of the sequence, capturing long-range dependencies in the data.
4. **Layer Normalization:** Normalizes the outputs of the attention layers to stabilize training. This step helps in maintaining the stability of the training process and improves convergence.
5. **Final Dense Layers:** Consists of several fully connected layers with ReLU activations and dropout for classification. These layers are responsible for mapping the high-dimensional representations learned by the attention layers to the output space, where each class (movie) is represented.

7.5. The algorithm

The algorithm for this research was developed in the PyCharm environment using the PyTorch library. The code is divided into three main parts:

7.5.1. Model Definition

The model we used is a Transformer Encoder designed for text classification, implemented using PyTorch. We adapted the code from Keras to PyTorch and customized it for our task of time series classification. This adaptation involved adjusting the attention layer size to match the number of voxels in the region of interest we are training on, and modifying the number of neurons in the final layer to correspond to the number of movie clips.

```
class TransformerEncoder(nn.Module):
    def __init__(self, num_voxels, classes, time2vec_dim, num_heads=1, head_size=128, ff_dim=None, num_layers=1,
                 dropout=0):
        super(TransformerEncoder, self).__init__()
        self.encoder_input_layer = nn.Linear(in_features=num_voxels, out_features=ff_dim)
        if ff_dim is None:
            ff_dim = head_size
        self.dropout = dropout
        self.classes = classes
        self.positional_encoding = PositionalEncoding(dropout=0.1, max_seq_len=seq_len, d_model=embedding_dim,  batch_first=False)
        self.attention_layers = nn.ModuleList(
            [AttentionBlock(num_heads=num_heads, head_size=head_size, ff_dim=ff_dim, dropout=dropout) for _ in range(num_layers)])
        self.norm = nn.LayerNorm(ff_dim)
        self.final_layers = nn.Sequential(
            nn.Flatten(),
            nn.Linear(ff_dim * seq_len, 512),
            nn.ReLU(),
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Linear(128, classes)
        )

    def forward(self, inputs):
        x = self.encoder_input_layer(inputs)
        x = self.positional_encoding(x)
        for attention_layer in self.attention_layers:
            x, attention_scores = attention_layer(x)
        x = self.norm(x)
        x = self.final_layers(x)
        return x
```

7.5.2. Data Preprocessing

Data Normalization

- **Z-Score Normalization:** The raw fMRI data, represented as blood-oxygen-level-dependent (BOLD) signals, undergoes z-score normalization. This process standardizes the data by subtracting the mean and dividing by the standard deviation along the specified axis. This ensures that the data has a mean of 0 and a standard deviation of 1, making it suitable for input into the model.

Data Slicing

- **Temporal Slicing:** The fMRI data, which is a time-series signal, is sliced based on specific intervals. These intervals could represent different phases of the movie-watching experience, such as the beginning, middle, or end of a movie clip, or the rest periods between clips.
- **Slicing Strategies:**
    - **Start:** Uses the first few TRs (Repetition Times) of the sequence.
    - **Middle:** Selects TRs from the middle of the sequence.
    - **End:** Focuses on the last few TRs of the sequence.
    - **All:** Uses the entire sequence, with options to pad shorter sequences with zeros or truncate longer ones.
- **Selection of TRs:** The chosen slice length is critical as it influences the amount of temporal information the model receives. For example, using the last 5 TRs (seconds) can provide a concise summary of neural activity towards the end of a stimulus.

```python
# Function to process movie data by slicing based on the configuration
def process_movie_data(data_vis, slice, task, inputs, outputs):
    for movie in range(1, 15):
        movie_data = data_vis[data_vis['y'] == movie]
        input = movie_data.iloc[:, :-3]
        if slice == 'start':
            input = input.iloc[:5, :]
        elif slice == 'end':
            input = input.iloc[-5:, :]
        elif slice == 'middle':
            start_index = len(input) // 2 - 7
            input = input.iloc[start_index:start_index + 5, :]
        elif slice == 'all':
            if task == 'movies':
                all_len = 260
            else:
                all_len = 20
            zeroes = pd.DataFrame(0, index=range(all_len - len(input)), columns=input.columns)
            input = pd.concat([input, zeroes], axis=0)
        else:
            raise Exception("For now you can choose slice from [start, middle, end, all]")
        if input.empty:
            print(f"Input data for movie {movie} is empty.")
            continue
        output = [0.0 for i in range(1, 15)]
        output[movie - 1] = 1.0
        inputs.append(torch.tensor(input.values))
        outputs.append(torch.tensor(output))
    return inputs, outputs
```

Synthetic Data Creation

- **Synthetic Subject Generation:** To augment the dataset, synthetic subjects are created by adding varying levels of Gaussian noise to the original data. For each original subject, multiple synthetic subjects are generated, each with a different noise level. This increases the overall dataset size and introduces variability, which can help the model generalize better.

Averaging Across Dimensions

- **Spatial and Temporal Averaging:** In some preprocessing steps, the data is averaged across spatial dimensions (e.g., across voxels in a specific brain region) or temporal dimensions (e.g., averaging across the last few TRs). This can reduce noise and highlight the most relevant signals in the data, improving the model's ability to learn meaningful patterns.

Grouping Subjects into Groups

Grouping subjects into groups of 4 serves as a method to reduce inter-subject variability and enhance the model's ability to detect consistent patterns across different individuals. By averaging the data across four subjects, the resulting "averaged subject" represents a more generalized brain response, which can improve the model's robustness and reduce the impact of outliers and noise. The steps are:

- **Random Selection:** Subject folders are randomly shuffled to ensure that the groups formed are not biased by any particular order.
- **Creation of Group Folders:** After shuffling, the subjects are divided into groups of four. For each group, a new "group folder" is created.
- **Averaging Data:** Within each group folder, the fMRI data from the four subjects is averaged. This involves combining the voxel data from each subject and computing the mean for each voxel across all subjects in the group. The averaged data is then treated as a single "synthetic subject" representing the group.
- **Grouping Logic:** This process is repeated across all available subject folders, ensuring that every subject is included in a group. The result is a set of group-level data that reduces individual variability and emphasizes consistent neural patterns.

Data Loading and Tensor Creation

- **Loading Subject Data:** The preprocessed data for each subject is loaded from `.pkl` files. These files contain the fMRI data along with associated metadata.
- **Tensor Conversion:** The data is then converted into PyTorch tensors. Inputs (the fMRI data) and outputs (the one-hot encoded labels for movies) are stacked into tensors, which are used to create PyTorch `TensorDataset` objects.
- **Dataloader Creation:** These datasets are then wrapped in PyTorch `DataLoader` objects, which handle batching and shuffling during training and evaluation.

Handling of Different Phases (Train, Eval, Test)

- The data preprocessing code is designed to handle different phases of model development—training, evaluation, and testing. Depending on the phase, different subsets of the data are loaded and processed. For instance, during training, data augmentation may be applied.

7.5.3. Main model

- **Main Loop:** Iterates over combinations of ROIs, Depending on the run_mode, it either performs cross-validation or calls the train function to execute normal training.

```
# Loop for running normal training or cross-validation based on configuration
for NET in NET_list:
    for NET_idx in NET_indexes:
        for H in H_list:
            if f'{NET}_{NET_idx}_{H}' not in exists_list:
                continue
            print(
            f"Running training on {H}_{NET}_{NET_idx} for {epochs} epochs - Batch Size: {batch_size}, Learning Rate: {learning_rate}")
            if run_mode == 'cross_val':
                run_cross_validation()
            else:
                train()
```

- **The train function:** responsible for setting up the overall training environment. It initializes the model, configures data loaders, and logs the experiment details with WandB. The function ensures that the data loaders are valid and then delegates the actual training process to the train loop.

- **The train loop function:** handles the core of the training and validation process. It iteratively trains the model over multiple epochs, adjusts learning rates using a scheduler, and logs metrics at regular intervals. During each epoch, the loop computes the training loss, updates the model parameters through backpropagation, and evaluates the model on the validation set. The train loop monitors the model's performance, saving the best-performing model based on validation accuracy to ensure that the most effective model is retained for testing. Together, the train function and train loop ensure a structured, efficient, and well-documented training process.

```
def train():
    global num_heads
    inputs = []
    outputs = []

    train_dataloader, eval_dataloader, test_dataloader, num_voxels = get_dataloaders2(directory, NET, NET_idx, H,
                                                                                      slice, batch_size, task,
                                                                                      Avg=Avg, noise_level=noise_level, n_synthetic_TRs=n_synthetic_TRs,
                                                                                      Create_synthetic_subjects=Create_synthetic_subjects, n_synthetic_subjects=n_synthetic_subjects,
                                                                                      Group_by_subjects=Group_by_subjects, group_size=group_size,
                                                                                      use_original_for_val_test=use_original_for_val_test)

    if train_dataloader is None or eval_dataloader is None or test_dataloader is None:
        print(f"Skipping {H}_{NET}_{NET_idx} as files are missing.")
        sys.exit()

    wandb.login()
    timestamp = time.strftime("%d%m-%H%M")
    wandb.init(
        project="fmri_project",
        group='encoder_nets',
        name=f'{NET}_{NET_idx}_{H}_{timestamp}_avg_{Avg}',
        config={
            "learning_rate": learning_rate, "epochs": epochs, "batch_size": batch_size, "dropout": dropout,
            "loss": 'CE', "optimizer": 'Adam',
            'attention heads': num_heads,
            "embedding dim": embedding_dim
        }
    )

    model_path = f'models/best_model_{NET}_{NET_idx}_{H}.pth'
    train_loop(train_dataloader, eval_dataloader, test_dataloader, num_voxels, model_path, run_mode)
    wandb.finish()
```

- **Transfer learning:** We also conducted an experiment where we pre-trained the Transformer model exclusively on the movie-watching data and subsequently fine-tuned it on the rest-period data. This method was intended to leverage the rich neural patterns associated with movie stimuli to enhance the model's performance during rest periods. However, contrary to our expectations, this process yielded slightly worse results compared to our standard training and testing procedure. These findings suggest that the neural representations during rest periods might require direct training rather than relying on patterns learned during movie-watching.

- **Cross-validation:**

  We use cross-validation in subject grouping to ensure the model's performance is robust and generalizable across different groups. By dividing the data into four folds, training on three, testing on the remaining fold, and then rotating through all folds for four runs, this method prevents overfitting and assesses the effectiveness of averaging data from groups of four subjects. Cross-validation provides reliable performance metrics, even when the data is scarce (since the grouping process divides it by four), confirming that the model can accurately classify fMRI data across various combinations of subject groups, validating the grouping strategy.

```
# Cross-validation function
def run_cross_validation():
    phase_path = osp.join(directory, 'cross_val', task)
    subject_folders = sorted(glob.glob(phase_path + '/**'))

    # Remove existing SYNTH_ and GROUP_ directories before running cross-validation
    for folder in subject_folders:
        if 'SYNTH_' in folder or 'GROUP_' in folder:
            print(f"Removing synthetic/group subject folder: {folder}")
            for file in glob.glob(osp.join(folder, '*')):
                os.remove(file)
            os.rmdir(folder)
    subject_folders = [folder for folder in subject_folders if 'SYNTH_' not in folder and 'GROUP_' not in folder]

    data_files = [osp.join(folder, f'{H}_{NET}_Avg.pkl') for folder in subject_folders]

    kf = KFold(n_splits=k)
    fold_accuracies = []
    train_accuracies = []
```

```
for fold, (train_index, test_index) in enumerate(kf.split(data_files)):
    run_suffix = f'run_{fold + 1}'
    train_files = [data_files[i] for i in train_index]
    test_files = [data_files[i] for i in test_index]

    train_dataloader, num_voxels = get_dataloaders2_for_cross_val(train_files, NET, NET_idx, H, batch_size, slice, task, Avg,
                                                    noise_level, n_synthetic_TRs, Group_by_subjects, group_size, run_suffix)
    test_dataloader, _ = get_dataloaders2_for_cross_val(test_files, NET, NET_idx, H, batch_size, slice, task, Avg,
                                            noise_level, n_synthetic_TRs, Group_by_subjects, group_size, run_suffix)

    if train_dataloader is None or test_dataloader is None:
        print(f"Skipping fold {fold + 1} due to missing data.")
        continue

    model_path = f'models/best_model_fold_{fold + 1}.pth'
    fold_accuracy, train_loss, train_accuracy = train_loop(train_dataloader, None, test_dataloader, num_voxels, model_path, 'cross_val')

    fold_accuracies.append(fold_accuracy)
    train_accuracies.append(train_accuracy)
    print(f"Fold {fold + 1} - Train Accuracy: {train_accuracy}, Test Accuracy: {fold_accuracy}")

average_accuracy = sum(fold_accuracies) / len(fold_accuracies)
print(f'Average k-fold accuracy: {average_accuracy}')
wandb.log({'Average k-fold accuracy': average_accuracy})
average_train_accuracy = sum(train_accuracies) / len(train_accuracies)
print(f'Average train accuracy: {average_train_accuracy}')
wandb.log({'Average train accuracy': average_train_accuracy})
wandb.finish()
```

8. Experiments and Results

In this section, we present the detailed findings and performance evaluation of our transformer-based model applied to fMRI data derived from movie-watching scenarios. The evaluation encompasses various experiments conducted with different hyperparameters and datasets, focusing on metrics such as accuracy and loss.

Our results cover an in-depth analysis of all time points (TRs) within the movies dataset. Additionally, we conduct comparative analyses by examining specific subsets of TRs from different phases of the datasets. By comparing the last few TRs of the movie dataset with the initial and final TRs of the rest dataset, we aim to uncover the differences in neural activation patterns and memory processes under various conditions. This dual focus allows for a comprehensive understanding of how neural activity evolves over time and across different cognitive states.

8.1. Primary Results

Hyperparameters:

attention heads	1
Batch size	16
dropout	0.1
embedding dim	512
epochs	100
Learning rate*	0.0005
loss	"CE"
optimizer	"Adam"
padding	to 260

* Learning rate decay of 70% every 33 steps to reduce overfitting.

All TRs of Movies *dataset (260 TRs)*

ROI	**index**	**side**	**train/acc**	**eval/acc**	**test/acc**
pCunPCC	5	LH	1	0.9087	0.888
pCunPCC	6	LH	1	0.94	0.912
Default_temp	5	LH	1	0.849	0.936
Default_temp	6	LH	1	0.873	0.869
DorsAttn_Post	5	LH	1	0.88	0.897
DorsAttn_Post	6	LH	1	0.976	0.988
Vis	5	LH	1	0.726	0.773
Vis	6	LH	1	0.857	0.893
AVG				**0.8762125**	**0.8945**

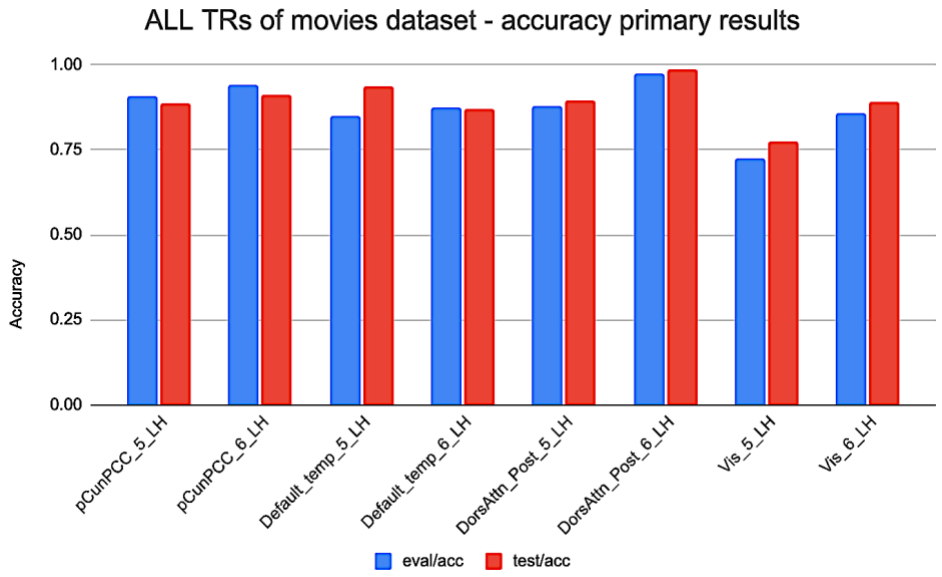

Figure 9 - Accuracy results of evaluation and test for all TRs of movies dataset across 8 ROIs.

Last 15 TRs results of movies dataset

- The padding created a bias in the results (as it was easier to classify by the number of zeroes added to each movie). We decided not to use any padding.
- Since later we would like to compare the movies analysis to the rest analysis, we had to decrease the number of TRs (as the rest segment only includes 20 TRs).
- The last TRs of the movies are better characterizing the movie representation.

ROI	train/acc	val/acc	test/acc
Vis_5_RH	0.8	0.68	0.68
Vis_5_LH	0.4	0.55	0.6
Vis_6_RH	0.4	0.19	0.2
Vis_6_LH	0.6	0.61	0.57
DorsAttn_Post_5_RH	0.2	0.4	0.34
DorsAttn_Post_5_LH	0.69	0.63	0.57
DorsAttn_Post_6_RH	0.2	0.31	0.22
DorsAttn_Post_6_LH	0.3	0.48	0.4
pCunPCC_5_RH	0.2	0.21	0.17
pCunPCC_5_LH	0.3	0.23	0.21
pCunPCC_6_RH	0.5	0.3	0.25
pCunPCC_6_LH	0.3	0.26	0.3
Default_PFC_1_LH	0.5	0.34	0.35
Default_PFC_2_LH	0.6	0.24	0.18
Default_temp_5_RH	0.25	0.39	0.38
Default_temp_5_LH	0.5	0.56	0.58
Default_temp_6_RH	0.13	0.19	0.19

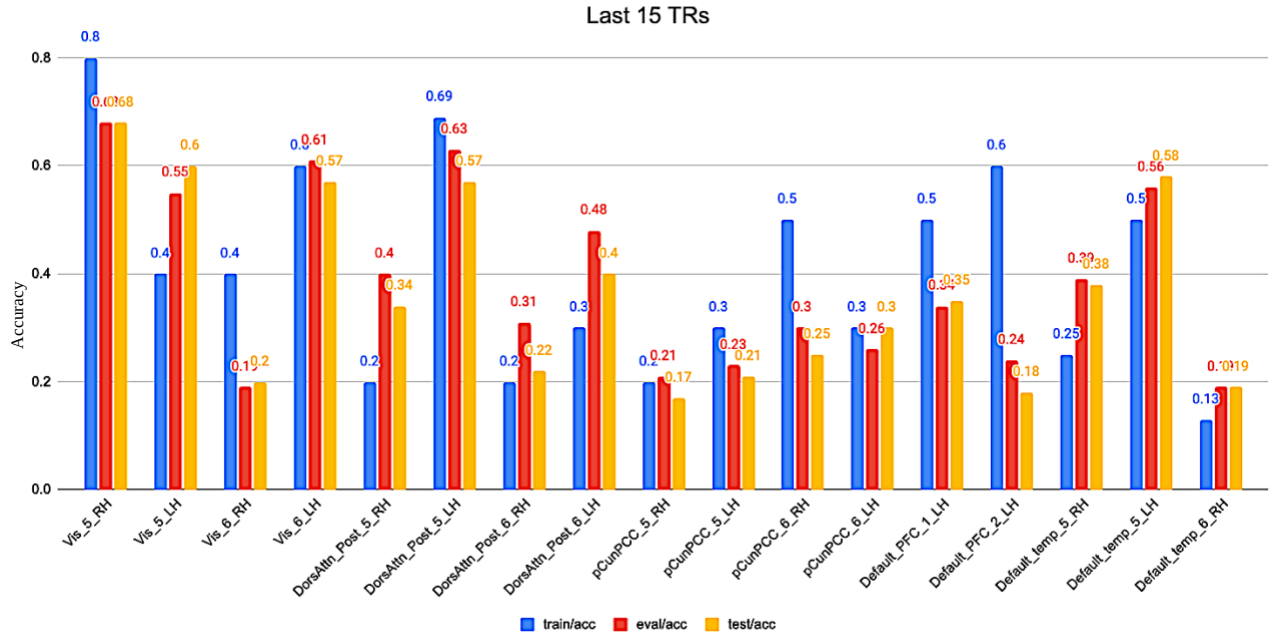

Figure 10 - Bar plot showing the accuracy of train, validation, and test for last 15 TRs of movies dataset across 17 ROIs

8.2. Results after hyperparameters tuning

Updated hyperparameters:

attention heads	4	We tried 1,2,4,8 attention heads, finding that 4 attention heads gave the highest accuracy.
Batch size	16	
dropout	0.5	We increased the dropout trying to reduce overfitting.
embedding dim	512	
epochs	30	Since we saw that 30 is enough.
Learning rate	0.00001	We tried different learning rates, finding that 0.00001 was the best.
loss	"CE"	
optimizer	"Adam"	
padding	no padding	

ALL TRs (65 TRs - the length of the shortest movie, no padding)

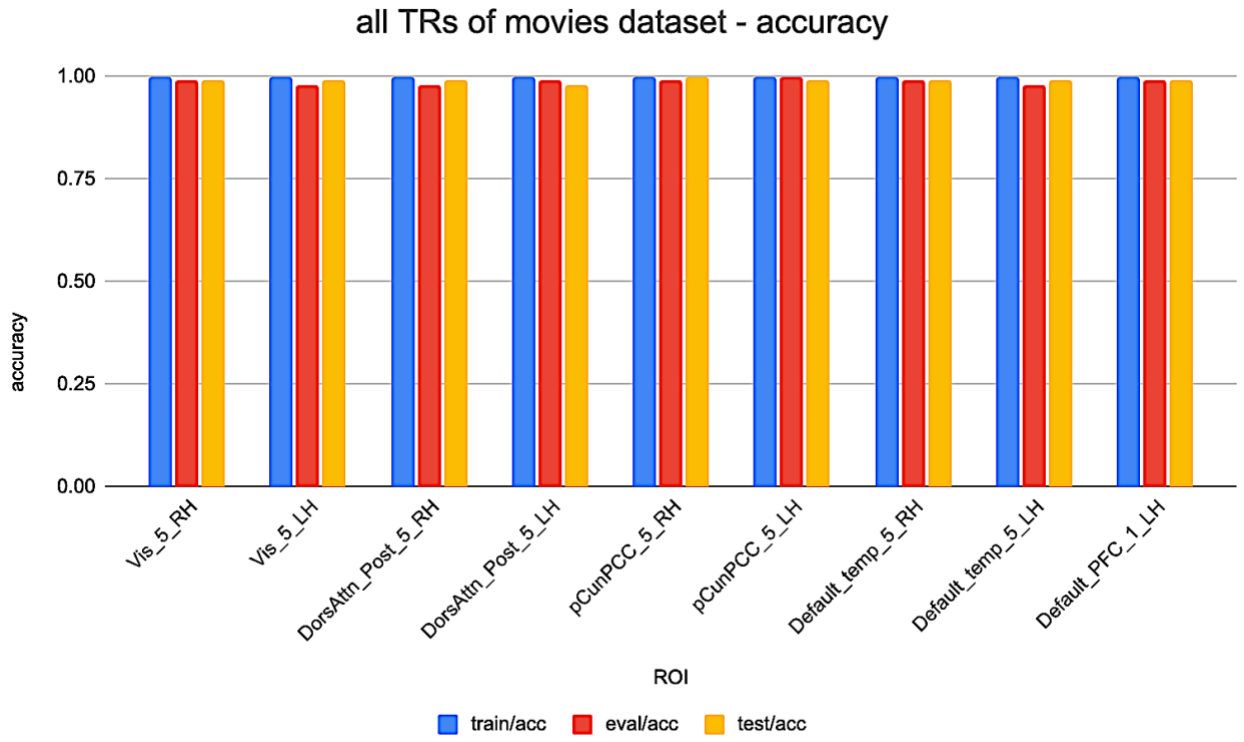

Figure 11 – Accuracy results for all TRs of movies dataset across 9 ROIs, after hyperparameters tuning

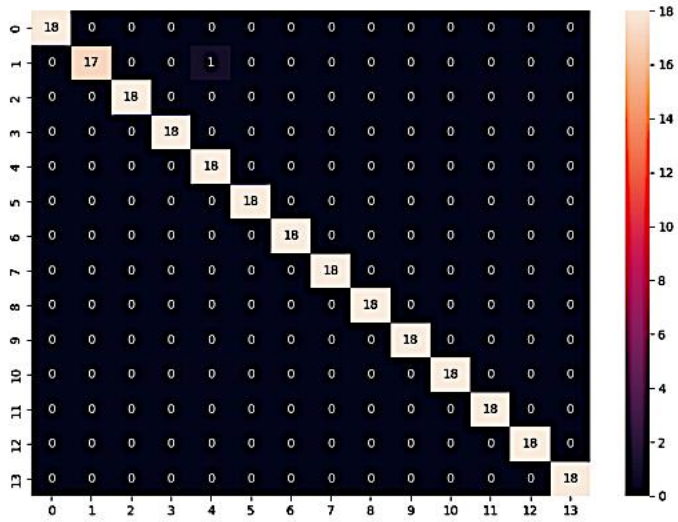

Figure 12 - Classification Heat Map for all TRs of movies dataset, Vis-5-RH area.

Last 15 TRs

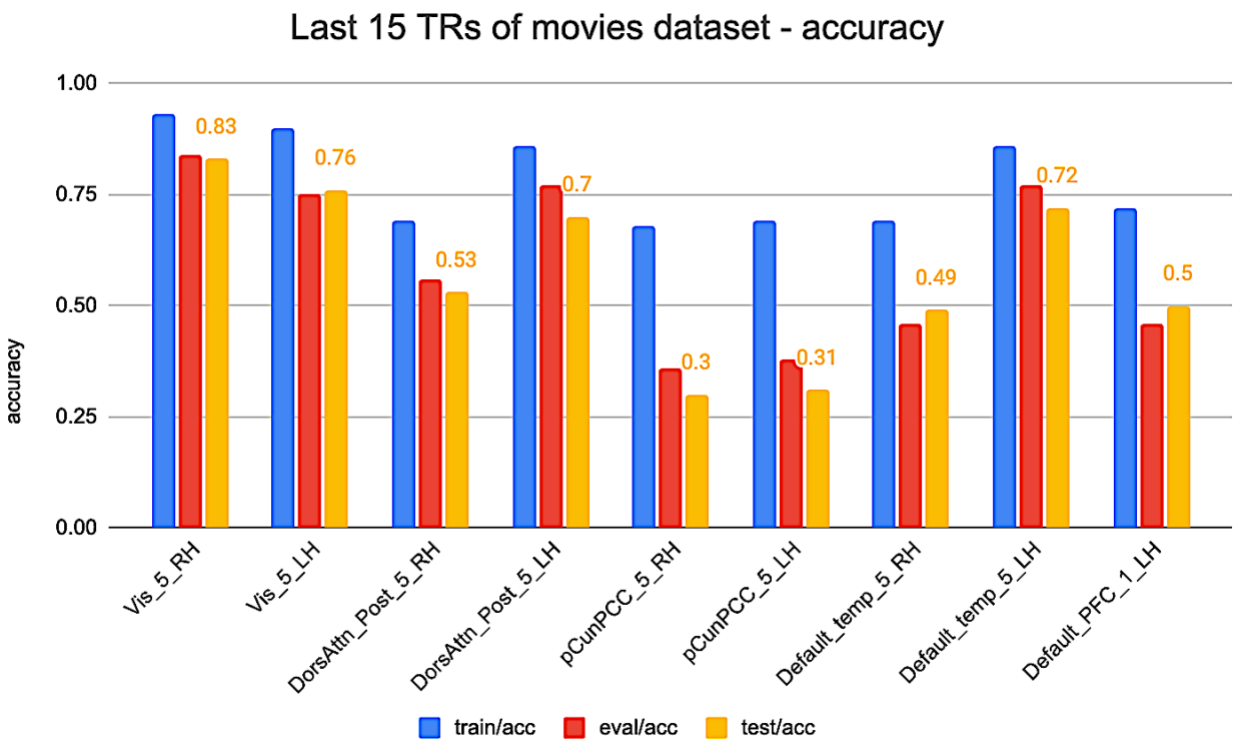

Figure 13 - Bar plot showing the accuracy of train, validation, and test for last 15 TRs of movies dataset across 9 ROIs.

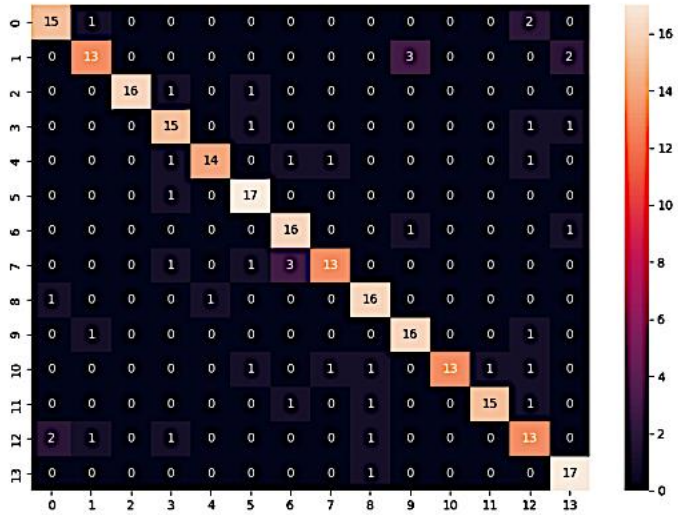

Figure 14 - Classification Heat Map for last 15 TRs of movies dataset, Vis-5-RH ROI.

Last 5 TRs

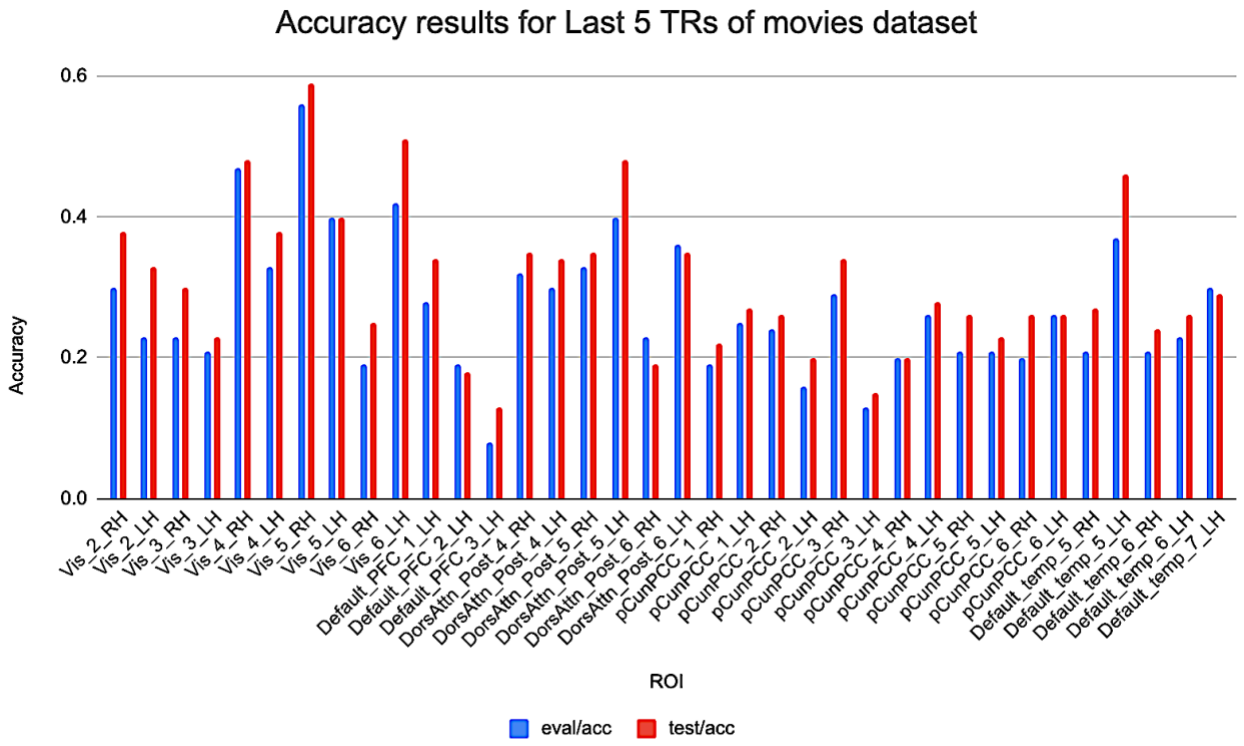

Figure 15 - Accuracy results for last 5 TRs of movies dataset.

8.3. Movies compared to Rest between movies analysis

- We wish to investigate memory traces after the movies in the rest period fMRI.
- We expect the last 5 TRs of the movie to show higher accuracy in classification than the first 5 TRs of the rest.
- We expect the last 5 TRs of the rest to have lower classification accuracy than the first 5 TRs of the rest.
- By comparing these three-time windows we wish to find that the rest segment contains memory traces of the movie, as it shows better accuracy than a random segment would.

Accuracy results of first and last 5 TRs of rest dataset

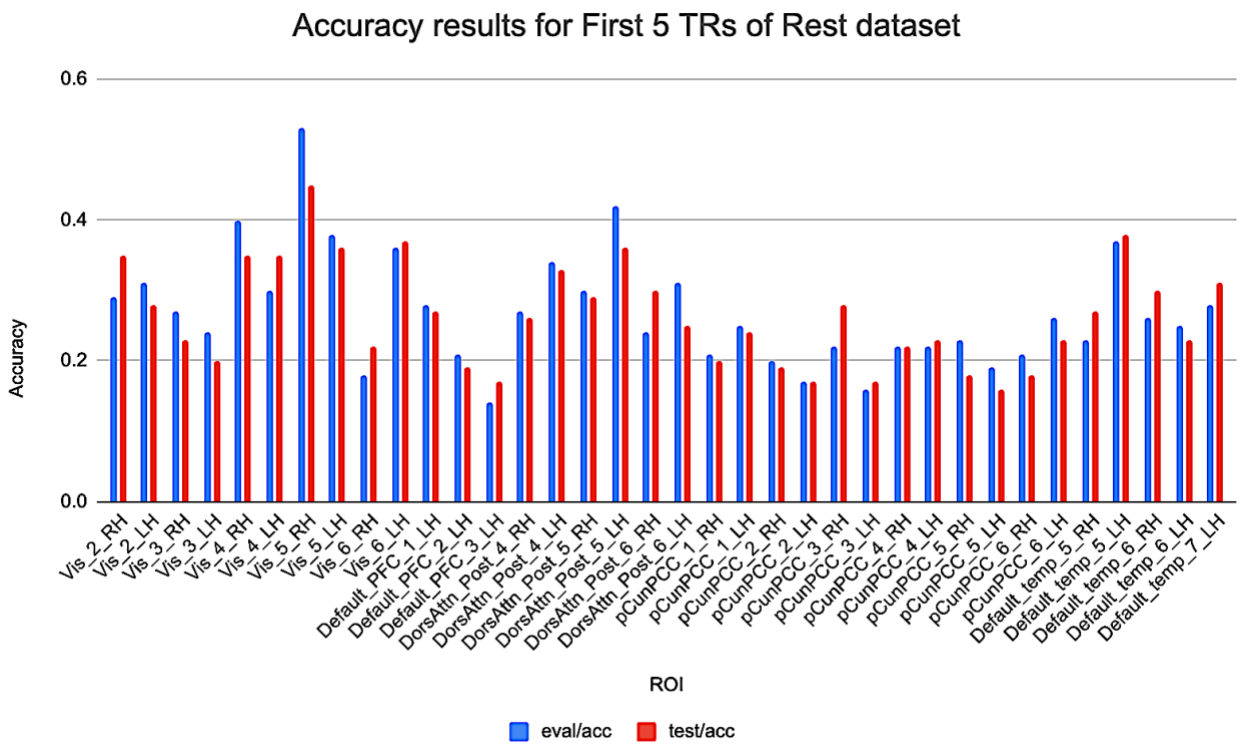

Figure 16 - Accuracy results for first 5 TRs of Rest dataset.

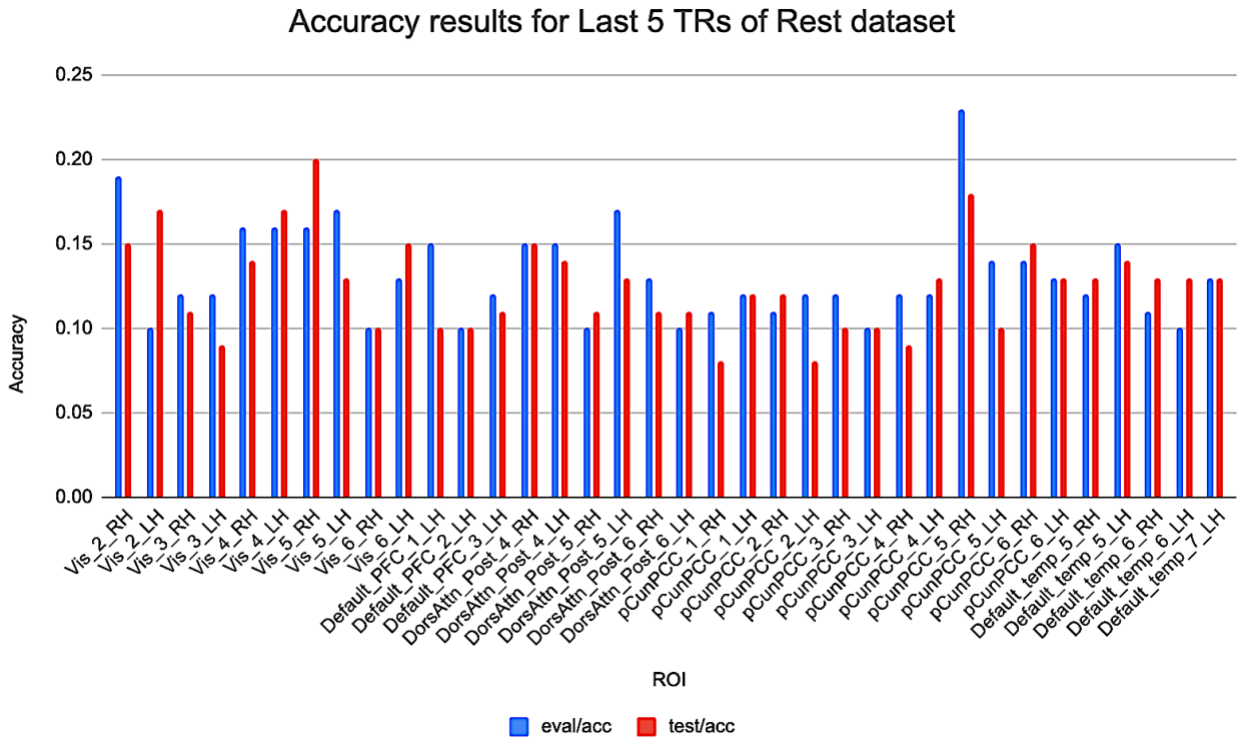

Figure 17 - Accuracy results for last 5 TRs of Rest dataset.

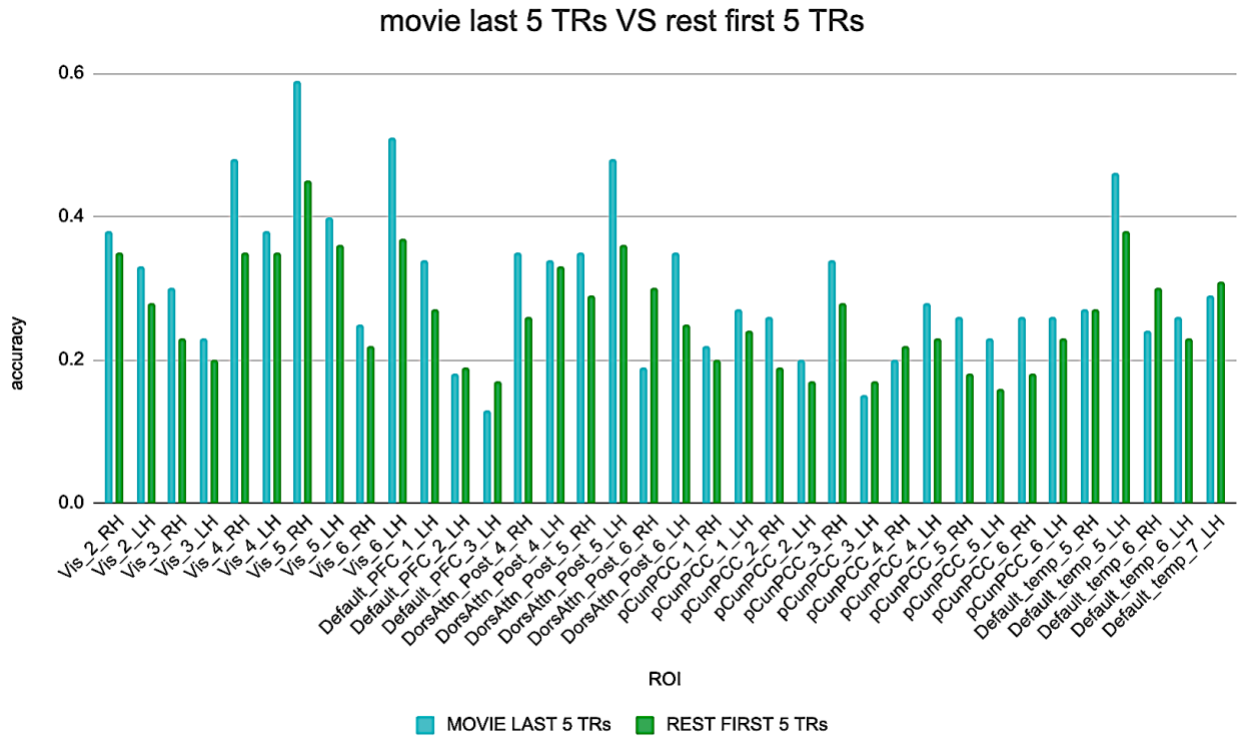

Figure 18 - Comparison between last 5 TRs of movies dataset and first 5 TRs of rest dataset, showing decrease from end of the movie to rest decreased of 13.5%.

Temporal averaging (a.k.a. "Averaging TRs"):
Every 5/10 TRs were averaged into 1 TR.

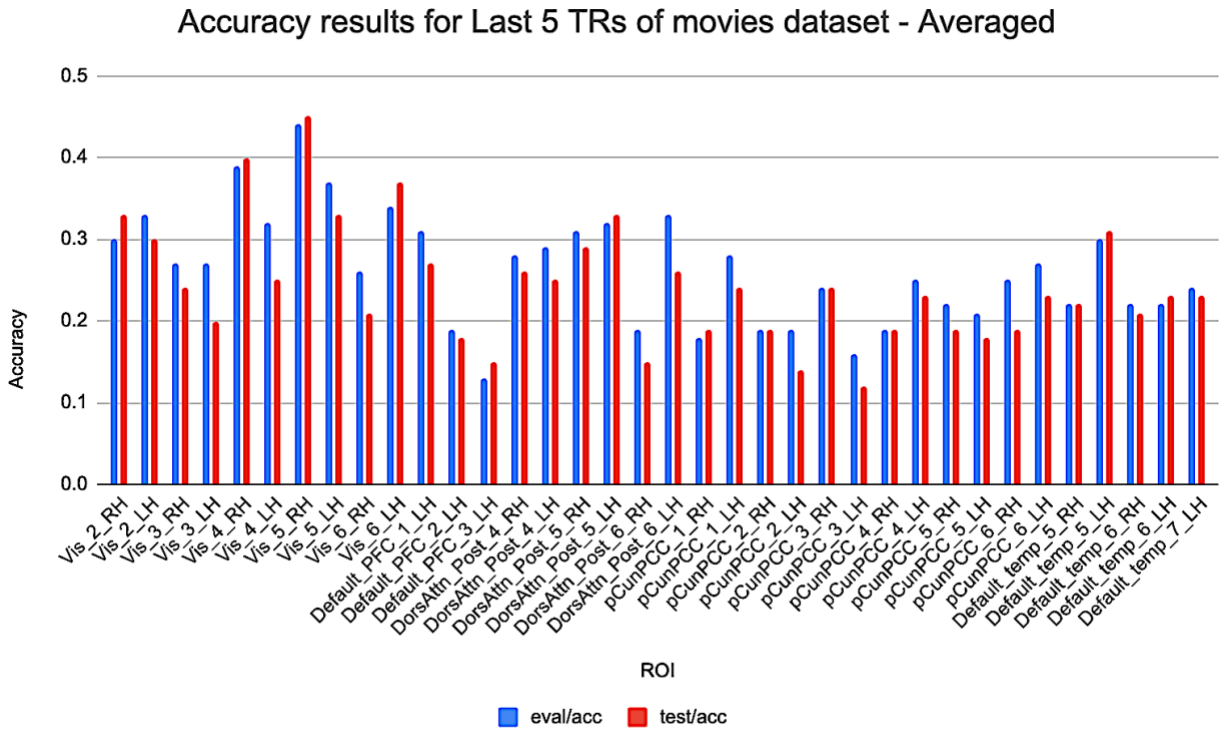

Figure 19 - Accuracy results for last 5 TRs of movies dataset - Averaged.

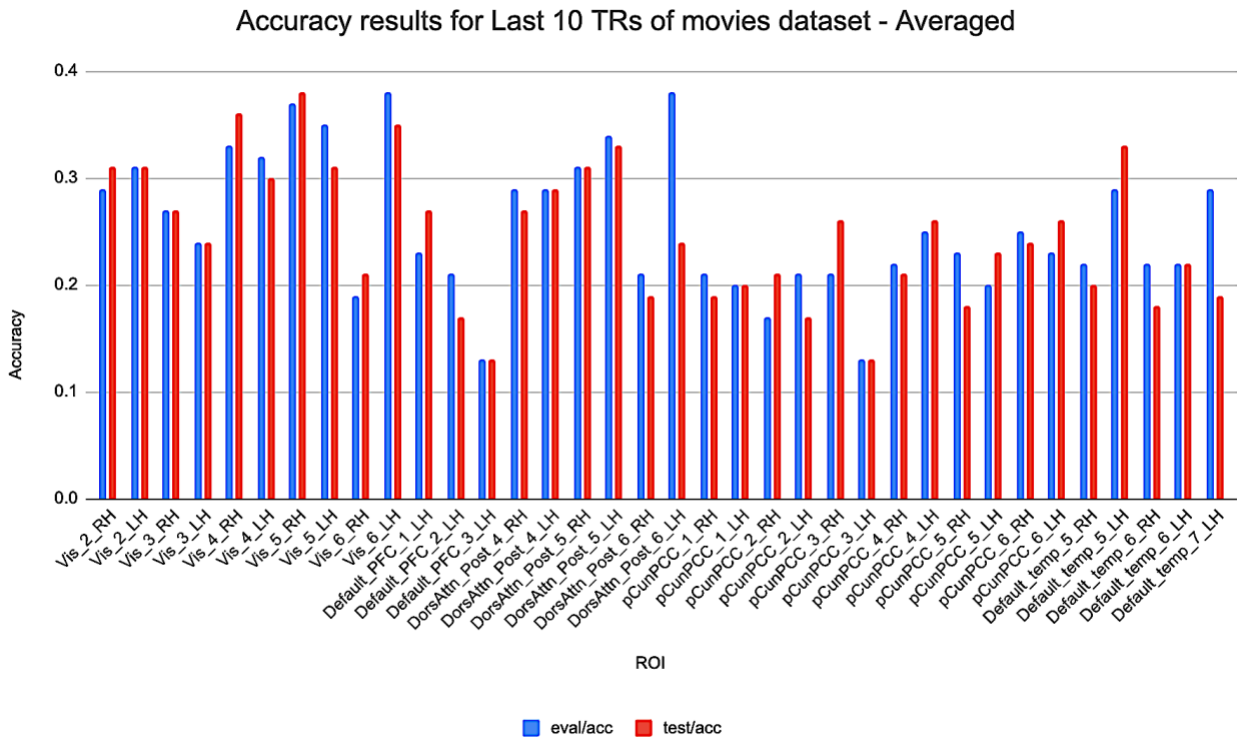

Figure 20 - Accuracy results for last 10 TRs of movies dataset - Averaged.

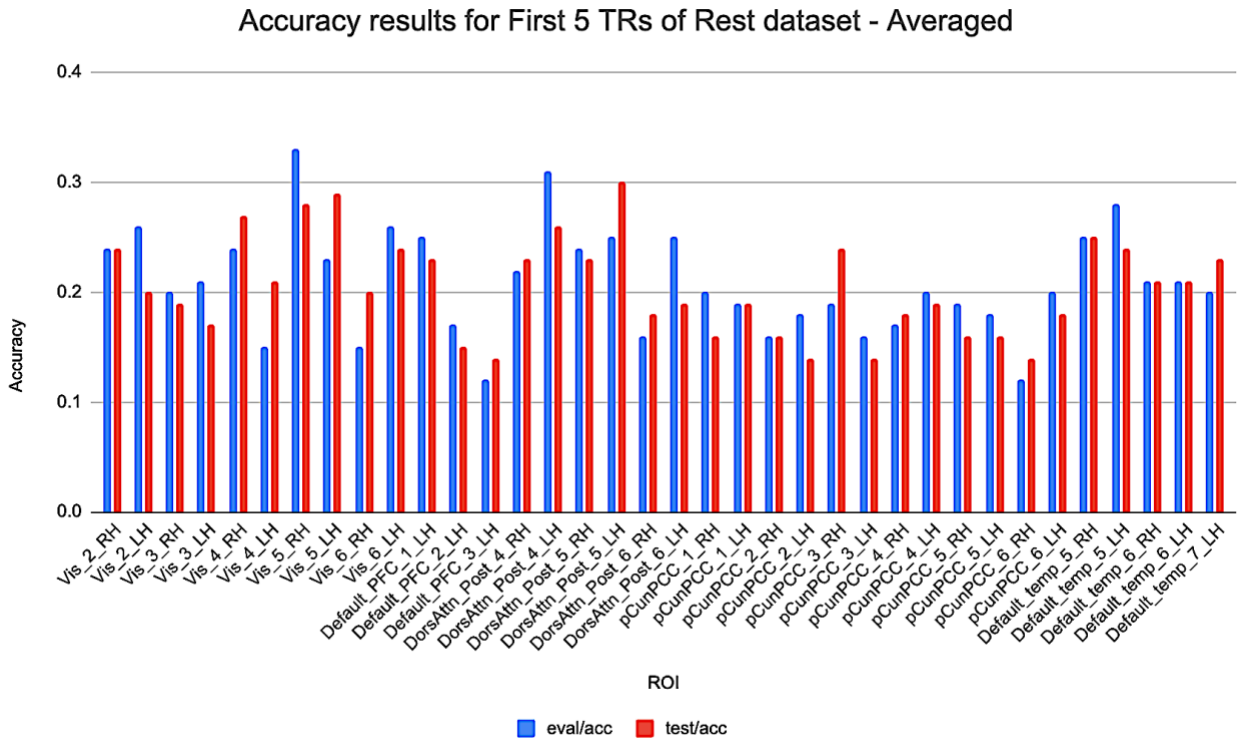

Figure 21 - Accuracy results for first 5 TRs of Rest dataset - Averaged.

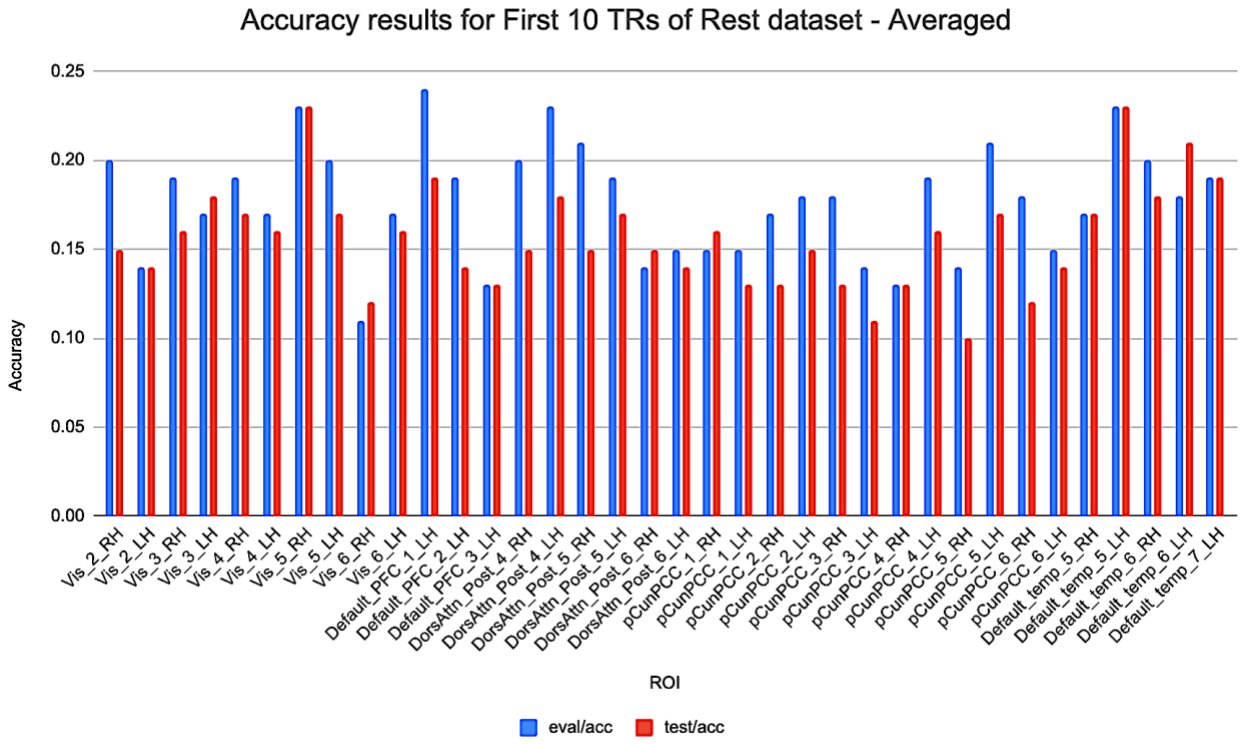

Figure 22 - Accuracy results for first 10 TRs of Rest dataset - Averaged.

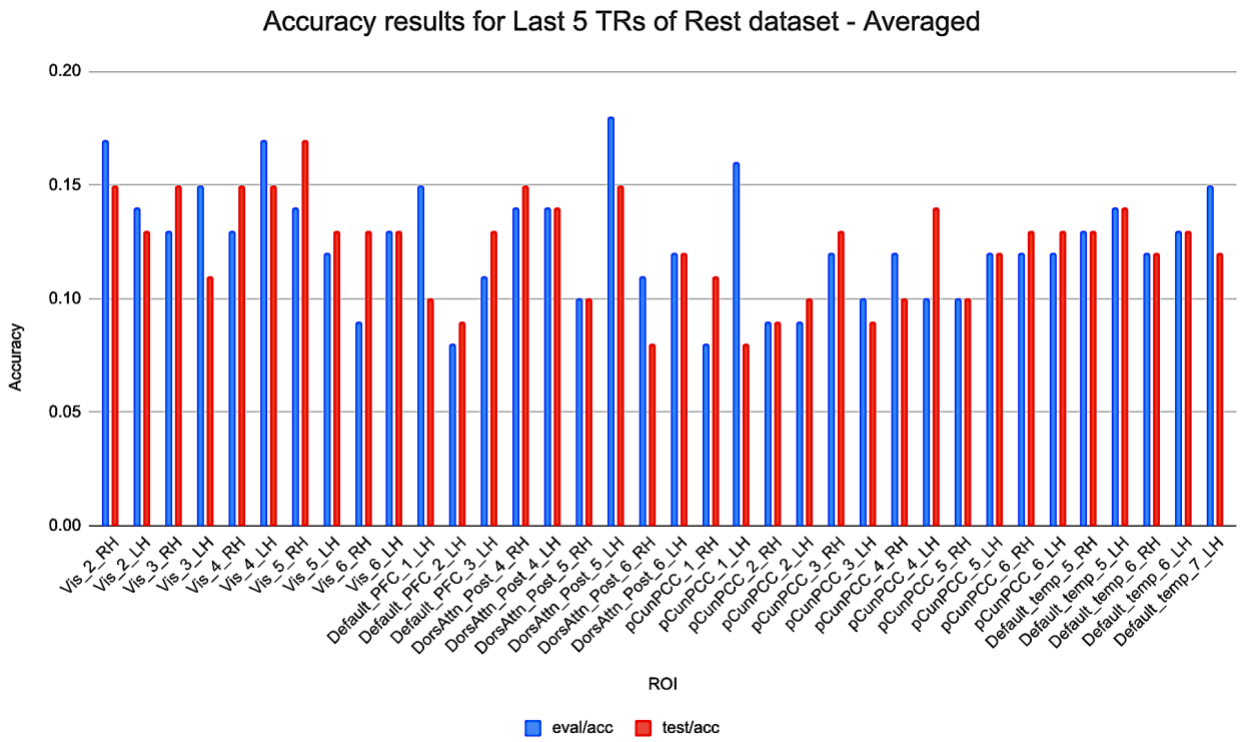

Figure 23 - Accuracy results for last 5 TRs of Rest dataset - Averaged.

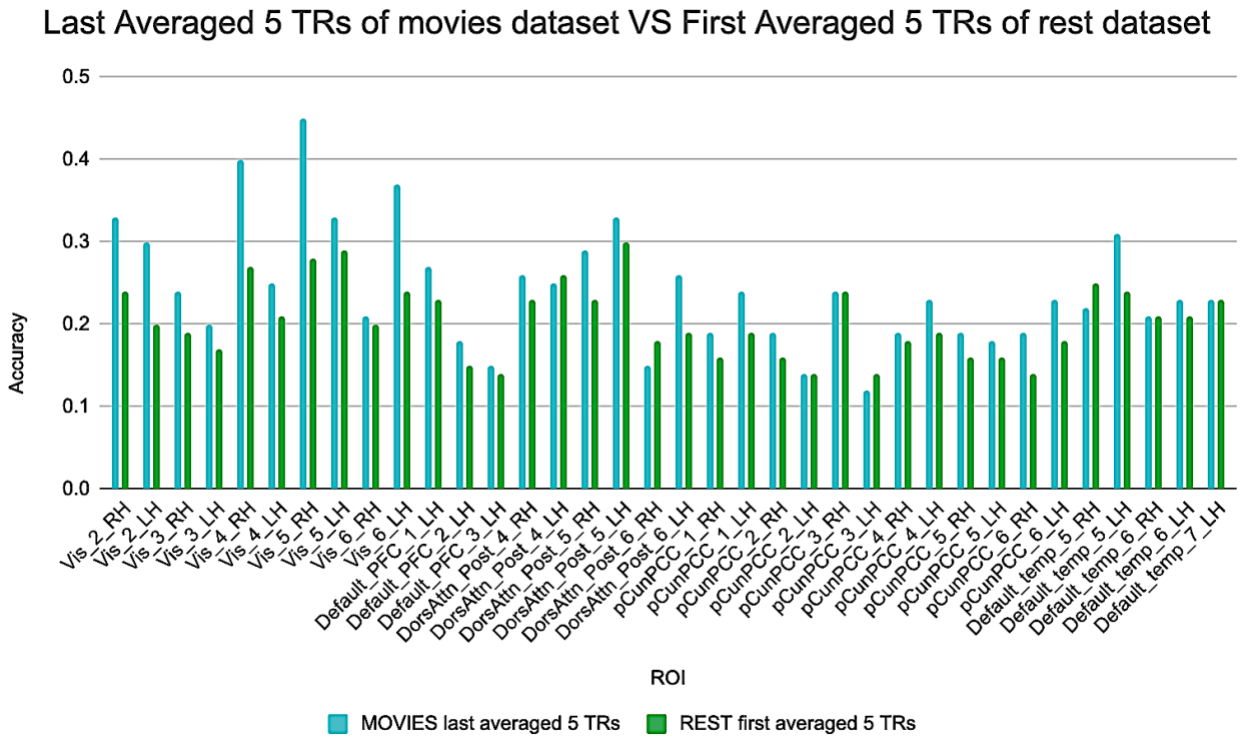

Figure 24 - Comparison between averaged last 5 TRs of movies dataset and averaged first 5 TRs of rest dataset, showing a decrease of 15.7% from end of the movie to beginning of rest.

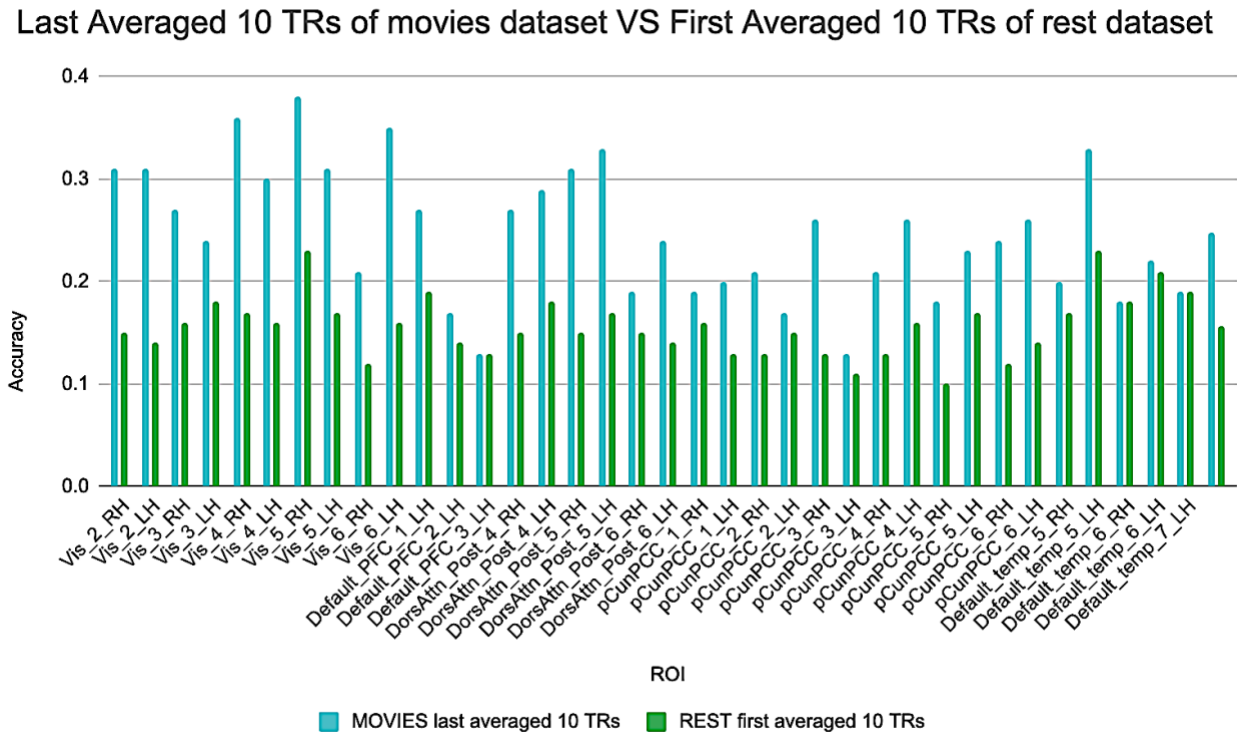

Figure 25 - Comparison between averaged last 10 TRs of movies dataset and averaged first 10 TRs of rest dataset - averaged, showing a decrease of 36.

8.4. Observation of a Specific ROI

In this section, we focus on the results of our experiments on a specific ROI to provide our chain of thoughts when designing the model. Specifically, we examine the Vis-5-RH ROI and present the curves of accuracy and loss over epochs, as well as the classification matrix. We chose this ROI because it shows the highest accuracy, as expected, since the visual network is highly involved when watching movies. This focused analysis allows us to gain deeper insights into the model's performance in a particular ROI.

We use the 5 TRs from the end of movies, the start of rest, and the end of rest for this detailed analysis. By analyzing the accuracy and loss curves, we can observe how the model's performance evolves during training and evaluation phases. The classification heat map provides a clear picture of how well the model performs in segmenting and classifying the data within this specific ROI. This targeted observation helps in understanding the nuances of the model's behavior and effectiveness in a localized context.

Curves of Accuracy/Loss and Heat Map of Vis-5-RH - Last 5 TRs of Movies Dataset:

summary metrics	
Eval/Accuracy	0.599
Eval/Loss	1.402
Test/Accuracy	**0.579**
Test/Loss	1.417
Train/Accuracy	0.658
Train/Loss	1.242

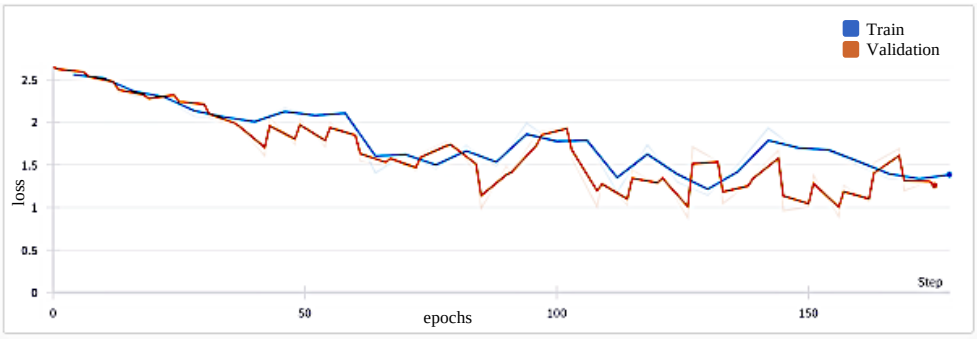

Figure 26 - Loss curves of validation and train for last 5 TRs of movies dataset, Vis-5-RH ROI.

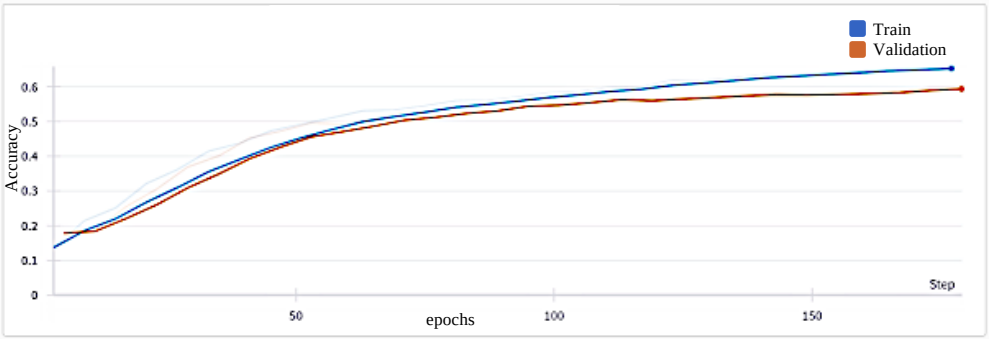

Figure 27 - Accuracy curves of validation and train for last 5 TRs of movies dataset, Vis-5-RH ROI.

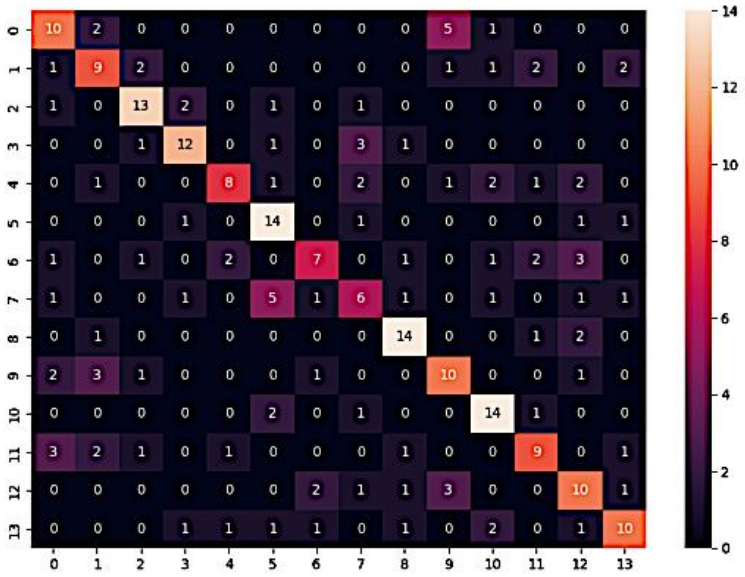

Figure 28 - Classification Heat Map for last 5 TRs of movies dataset, Vis-5-RH ROI.

Curves of Accuracy/Loss and Heat Map of Vis-5-RH - first 5 TRs of rest Dataset:

summary metrics	
Eval/Accuracy	0.515
Eval/Loss	1.33
Test/Accuracy	**0.472**
Test/Loss	1.644
Train/Accuracy	0.613
Train/Loss	1.362

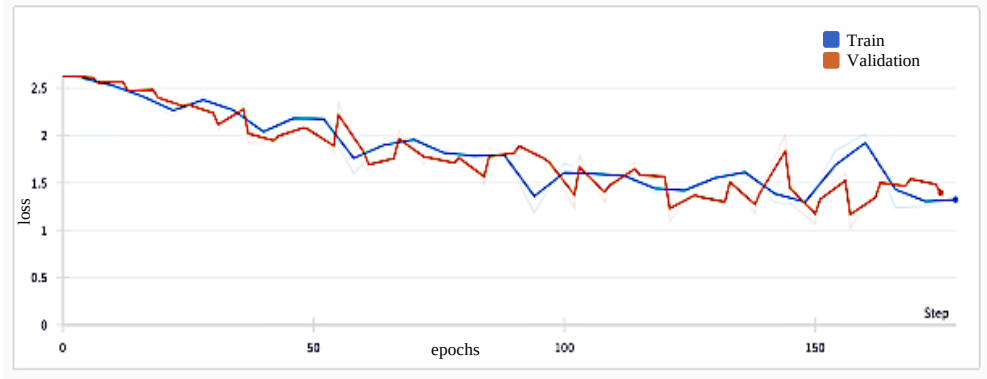

Figure 29 - Loss curves of validation and train for first 5 TRs of rest dataset, Vis-5-RH ROI.

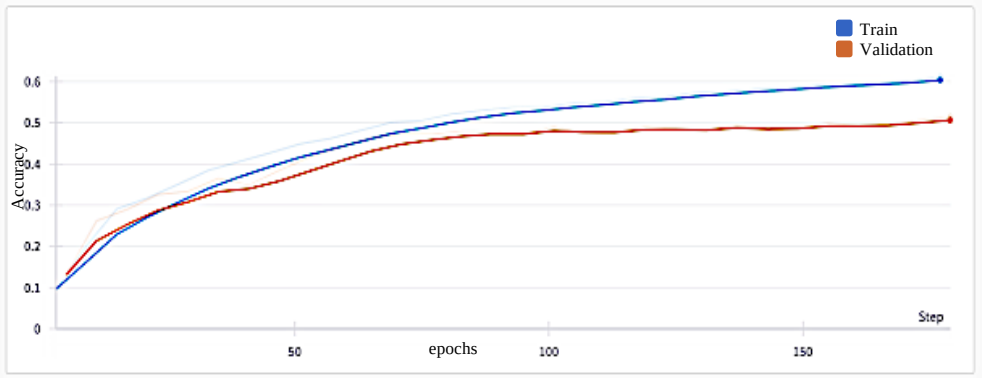

Figure 30 - Accuracy curves of validation and train for first 5 TRs of rest dataset, Vis-5-RH ROI.

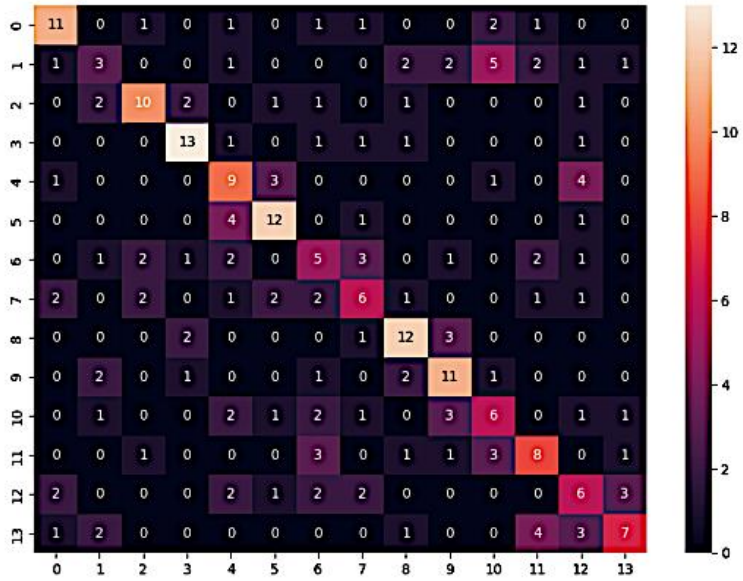

Figure 31 - Classification Heat Map for first 5 TRs of rest dataset, Vis-5-RH ROI.

Curves of Accuracy/Loss and Heat Map of Vis-5-RH - last 5 TRs of Rest Dataset:

summary metrics	
Eval/Accuracy	0.178
Eval/Loss	2.539
Test/Accuracy	**0.202**
Test/Loss	2.487
Train/Accuracy	0.374
Train/Loss	2.119

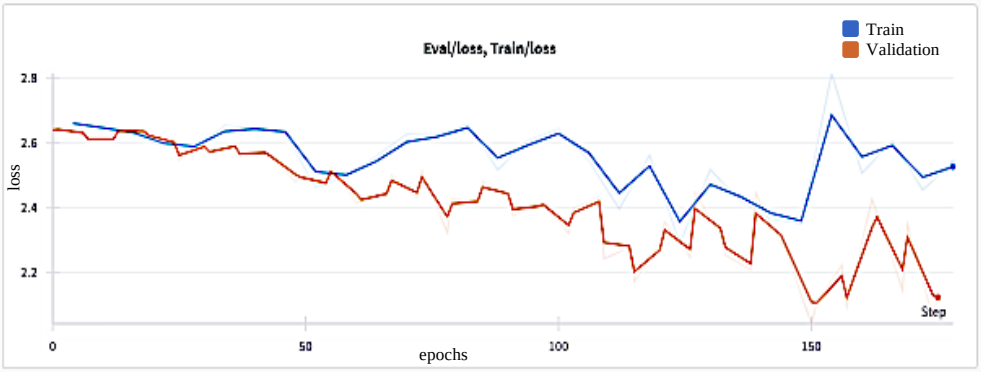

Figure 32 - Loss curves of validation and train for last 5 TRs of rest dataset, Vis-5-RH ROI.

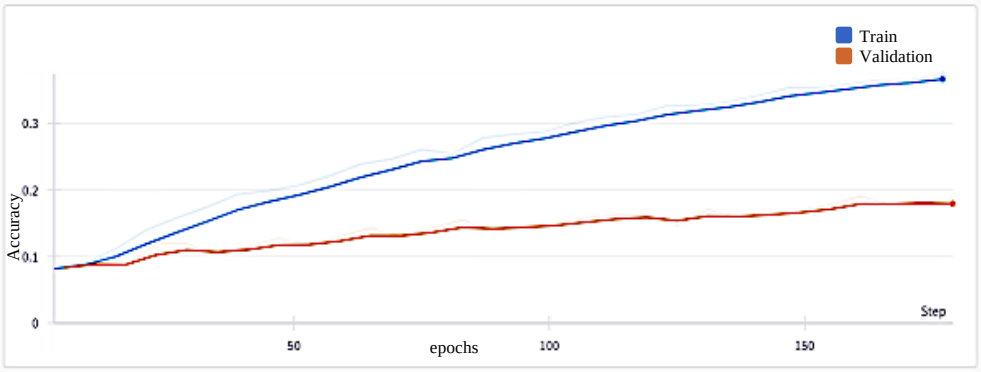

Figure 33 - accuracy curves of validation and train for last 5 TRs of rest dataset, Vis-5-RH ROI.

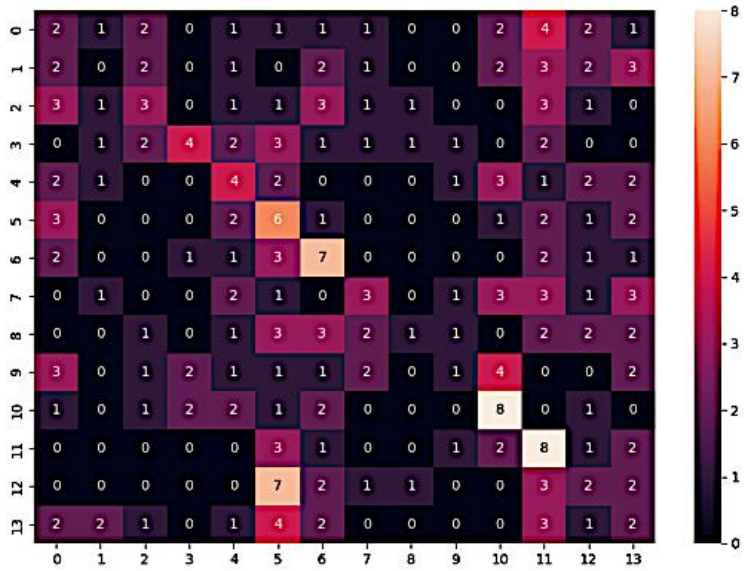

Figure 34 - Classification Heat Map for last 5 TRs of rest dataset, Vis-5-RH ROI.

8.4.1. Temporal Averaging (a.k.a. "Averaging TRs") – individual ROI and network comparison

We aimed to smooth the data by averaging the values across the last 5 TRs during the rest periods. This approach reduces noise and captures a more stable baseline activity level in the brain. Specifically, for every 5 TRs, we average the BOLD signal so that each voxel will have one averaged BOLD value (instead of 5). This method should reduce the variability within the rest periods and provide a clearer and more consistent signal.

Illustration:

Before Averaging TRs:

TR\voxels	voxel 1	...	...	voxel 160
1	bold value	bold value	bold value	bold value
2	bold value	bold value	bold value	bold value
3	bold value	bold value	bold value	bold value
4	bold value	bold value	bold value	bold value
5	bold value	bold value	bold value	bold value

After Averaging TRs:

TR\voxels	voxel 1	...	...	voxel 160
Averaged TR	bold value	bold value	bold value	bold value

Net averaging TRs:
concatenate all individual Vis ROIs to one file that includes all voxels

	ROI1				**ROI2**		
TR\voxels	voxel 1	...	...	voxel 160	voxel 1	...	voxel 140
Averaged TR	bold value	bold value	bold value	bold value	bold value	bold value	bold value

Results:

Due to the high overfitting observed before implementing TR averaging, we decided to adjust the hyperparameters to improve the model's generalization and reduce overfitting. The changes in hyperparameters are detailed in the following table:

attention heads	4	same
Batch size	16	same
dropout	0.7	We increased the dropout trying to reduce overfitting.
embedding dim	512	same
epochs	55	Epochs increased from 30 since we increased the LR
Learning rate	0.000001	Reduced from 0.00001
loss	"CE"	same
optimizer	"Adam"	same
padding	no padding	same
Weight decay	0.01	We used weight decay for L2 regularization and incorporated it into the optimizer to reduce overfitting.

Results: Individual ROI, averaged last 5 TRs of rest (vis_5_RH)

summary metrics	
Eval/Accuracy	0.095
Eval/Loss	2.616
Test/Accuracy	**0.095**
Test/Loss	2.630
Train/Accuracy	0.096
Train/Loss	2.640

Evaluations for the averaged last 5 TRs of rest dataset for the Vis-5-RH ROI

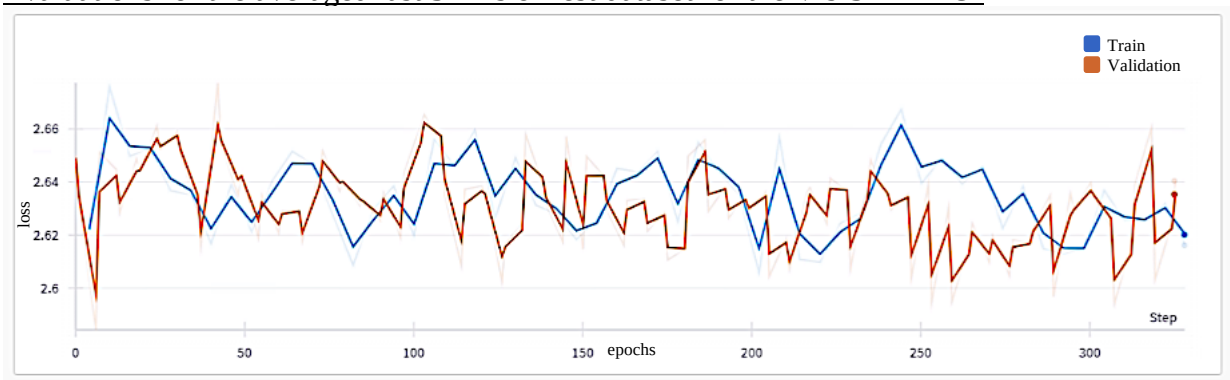

Figure 35 - training and validation loss curves for the averaged last 5 TRs of rest dataset, Vis-5-RH ROI.

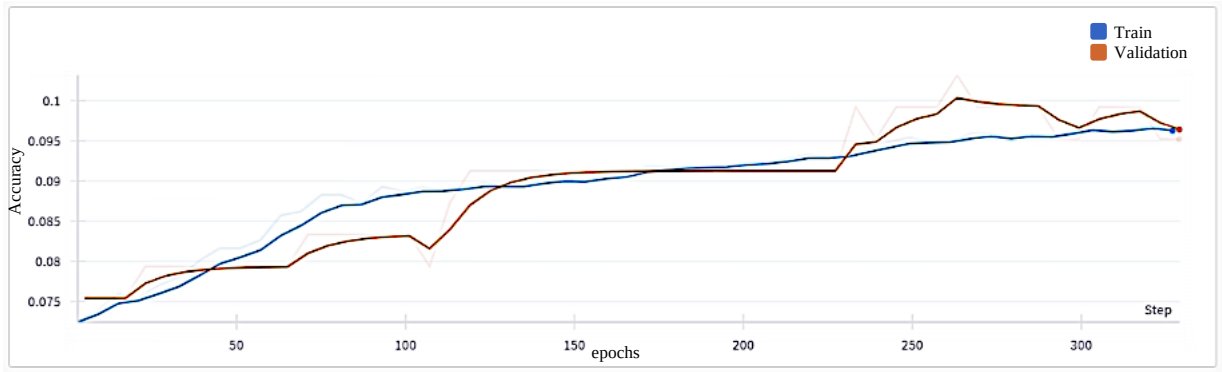

Figure 36 - training and validation accuracy curves for the averaged last 5 TRs of rest dataset, Vis-5-RH ROI.

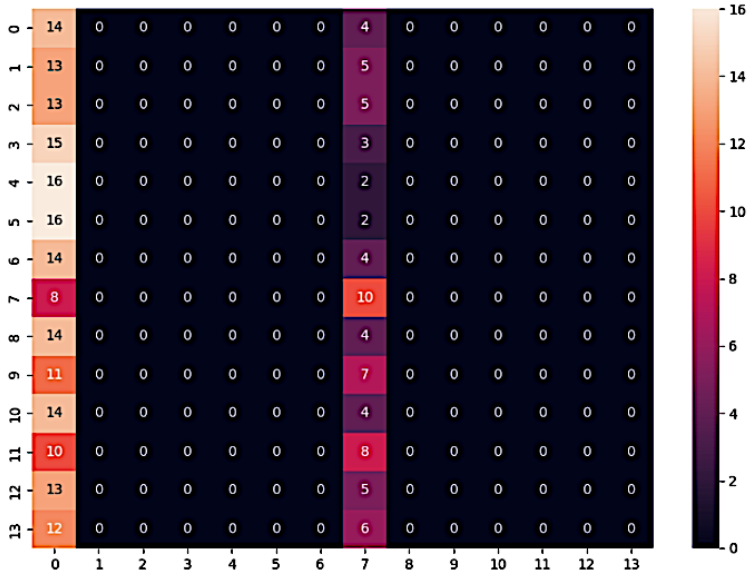

Figure 37 - Classification Heat Map for averaged last 5 TRs of rest dataset, Vis-5-RH ROI.

Evaluations for the averaged last 5 TRs of rest dataset for the Vis Network

summary metrics	
Eval/Accuracy	0.135
Eval/Loss	2.606
Test/Accuracy	**0.143**
Test/Loss	2.615
Train/Accuracy	0.184
Train/Loss	2.590

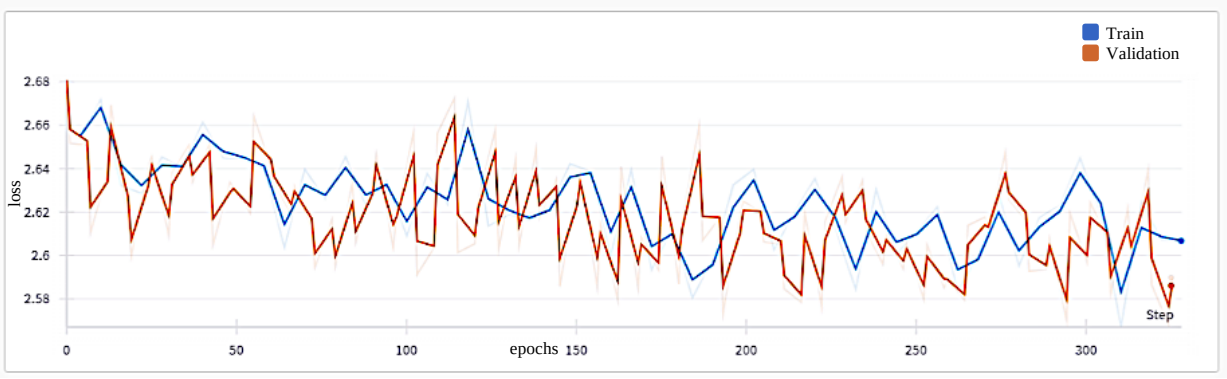

Figure 38 - training and validation loss curves for the averaged last 5 TRs of rest dataset, Vis Network

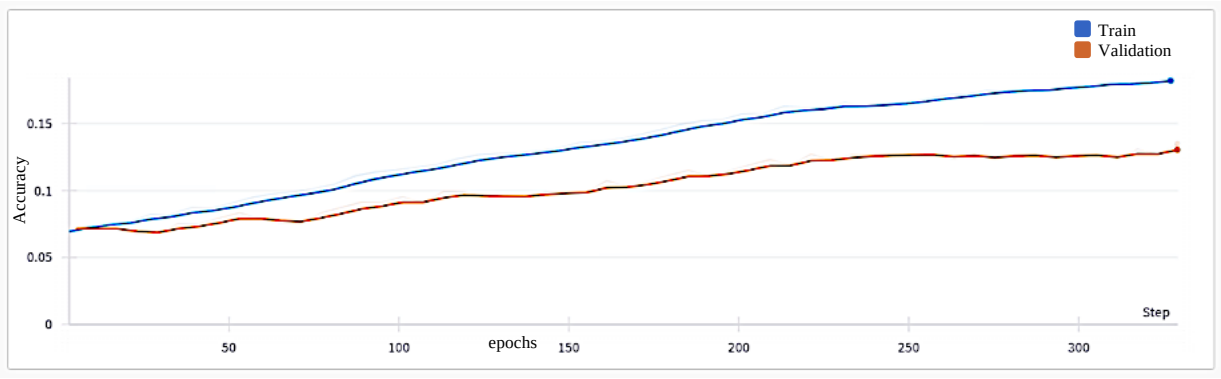

Figure 39 - training and validation accuracy curves for the averaged last 5 TRs of rest dataset, Vis Network

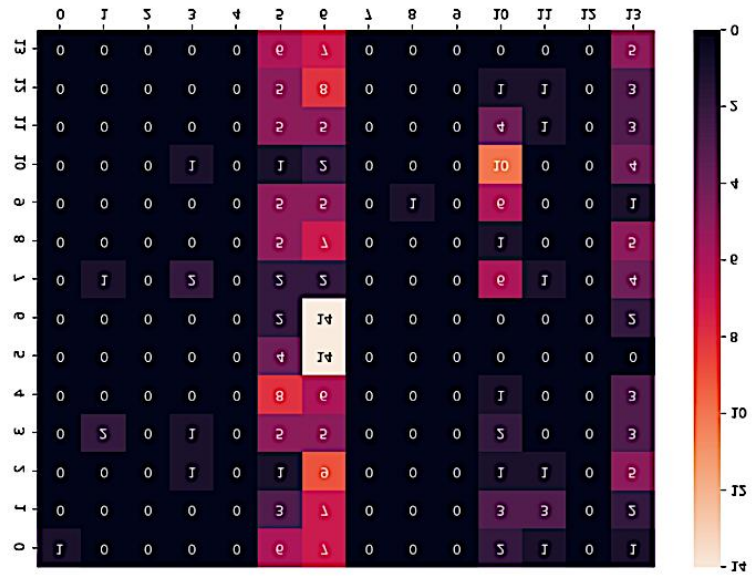

Figure 40 - Classification Heat Map for averaged last 5 TRs of rest dataset, Vis Network

8.4.1.1. Principal Component Analysis (PCA)

To address the high dimensionality of the data resulting from concatenating voxel data across rows, we applied PCA after averaging the TRs. PCA was utilized to reduce dimensionality while preserving the

most significant patterns in the data, thereby minimizing noise and redundancy. We selected 10 principal components for the analysis, capturing the key variance within the data.

The results of PCA for the averaged last 5 TRs of rest, Vis-5-RH:

summary metrics	
Eval/Accuracy	0.087
Eval/Loss	2.645
Test/Accuracy	**0.075**
Test/Loss	2.639
Train/Accuracy	0.085
Train/Loss	2.640

8.4.1.2. Data Augmentation of subjects

We introduced synthetic subjects to augment the training dataset (and by this, trying to decrease the overfitting we encountered) and improve the model's robustness. The creation of these synthetic subjects involved applying additive Gaussian noise to the original fMRI data. This noise was carefully calibrated to preserve the underlying distribution of signals both over time and over space.

For each original subject, we generated 10 synthetic versions by applying varying levels of noise. The noise levels ranged from 0.01 for the first synthetic subject to 0.1 for the tenth synthetic subject. The rationale for using incrementally increasing noise was to introduce a controlled variation in the data, simulating different levels of signal variability while still preserving the core characteristics of the original data. This approach allowed the model to learn from a broader spectrum of data, ranging from nearly identical to moderately altered versions of the original signals.

These synthetic subjects were used exclusively in the training phase, resulting in a training set that was multiplied by 11 (original subject, plus 10 synthetic subjects per each original one). This augmentation strategy aimed to enhance the model's ability to generalize to new, unseen data by providing it with a more diverse set of training examples.

Augmentation of subjects and averaging last 5 TRs of rest, Vis-5-RH:

summary metrics	
Eval/Accuracy	0.175
Eval/Loss	2.475
Test/Accuracy	**0.218**
Test/Loss	2.468
Train/Accuracy	0.226
Train/Loss	2.512

Augmentation of subjects and averaging last 5 TRs of rest, Vis network:

summary metrics	
Eval/Accuracy	0.286
Eval/Loss	2.219
Test/Accuracy	**0.302**
Test/Loss	2.143
Train/Accuracy	0.592
Train/Loss	1.256

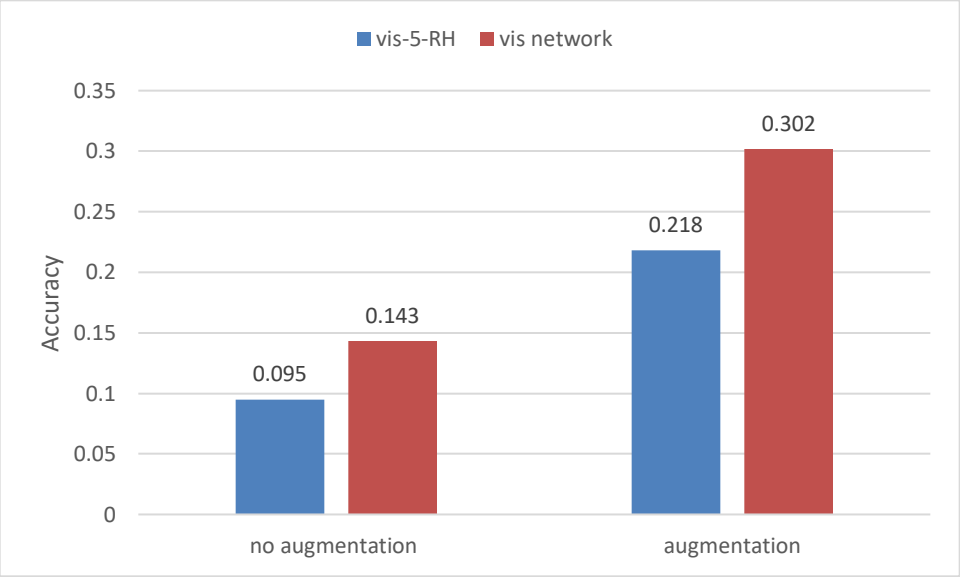

Figure 41 – Test accuracy of averaging TRs, before and after augmentation, vis-5-RH ROI vs Vis network.

8.4.1.3. Spatial Averaging (a.k.a. "Averaging Voxels") – individual ROI and network comparison

To explore another method of data smoothing, we averaged the BOLD signals across voxels within each ROI, providing a single representative value per TR. This experiment aims to reduce variability within the ROI and focus on the overall activity pattern.

For each TR, we calculated the average BOLD signal across all voxels within each ROI. This yielded one averaged BOLD value per TR for each ROI.

Illustration:

Before averaging voxels:

	ROI1				**ROI2**		
TR\voxels	voxel 1	...	...	voxel 160	voxel 1	...	voxel 140
TR1	bold value	bold value	bold value	bold value	bold value	bold value	bold value
TR2	bold value	bold value	bold value	bold value	bold value	bold value	bold value
…	bold value	bold value	bold value	bold value	bold value	bold value	bold value
TR 3229	bold value	bold value	bold value	bold value	bold value	bold value	bold value

After averaging voxels:

TR\voxels	**ROI1**	**ROI2**
TR 1	bold value	bold value
TR 2	bold value	bold value
....	bold value	bold value
TR 3229	bold value	bold value

Evaluations averaging voxels for the last 5 TRs of rest dataset for the Vis-5-RH ROI

summary metrics	
Eval/Accuracy	0.123
Eval/Loss	2.607
Test/Accuracy	**0.083**
Test/Loss	2.619
Train/Accuracy	0.116
Train/Loss	2.698

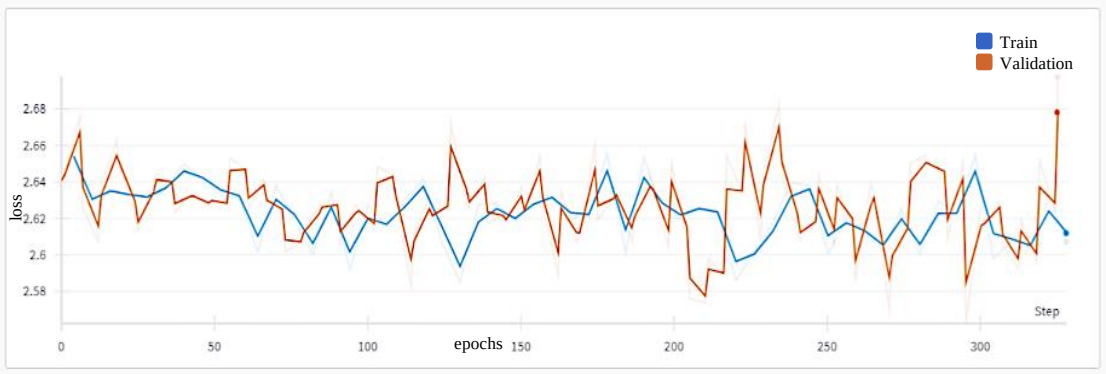

Figure 42 - training and validation loss curves for averaged voxels, last 5 TRs of rest dataset, Vis-5-RH ROI

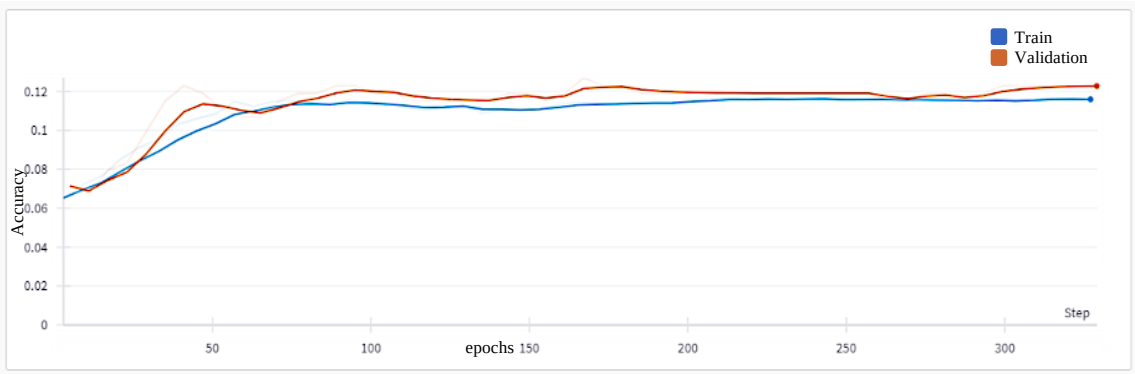

Figure 43 - training and validation accuracy curves for averaged voxels, last 5 TRs of rest dataset, Vis-5-RH ROI

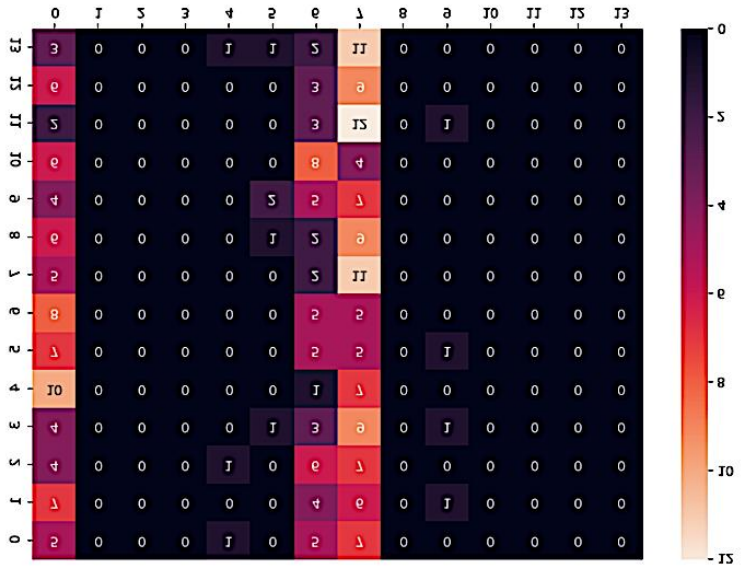

Figure 44 - Classification Heat Map for averaged voxels, last 5 TRs of rest dataset, Vis-5-RH ROI

Evaluations averaging voxels for the last 5 TRs of rest dataset for the Vis network

summary metrics	
Eval/Accuracy	0.151
Eval/Loss	2.557
Test/Accuracy	**0.143**
Test/Loss	2.615
Train/Accuracy	0.152
Train/Loss	2.601

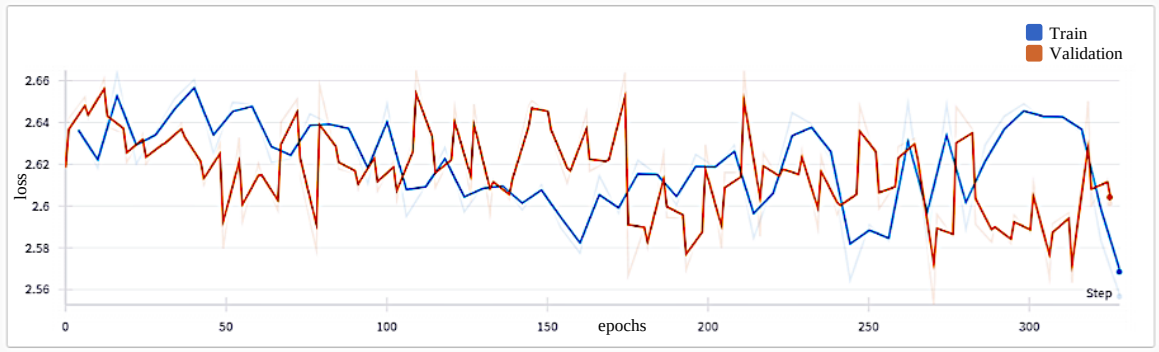

Figure 45 - training and validation loss curves for averaged voxels, last 5 TRs of rest dataset, Vis network

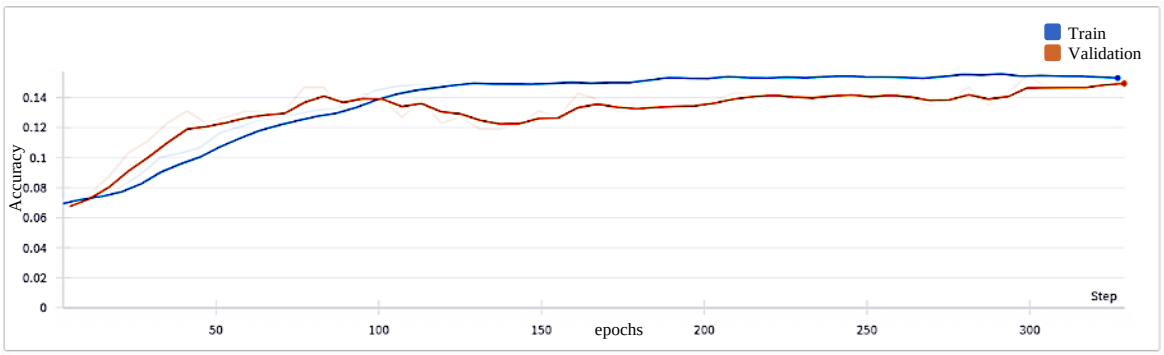

Figure 46 - training and validation accuracy curves for averaged voxels, last 5 TRs of rest dataset, Vis network

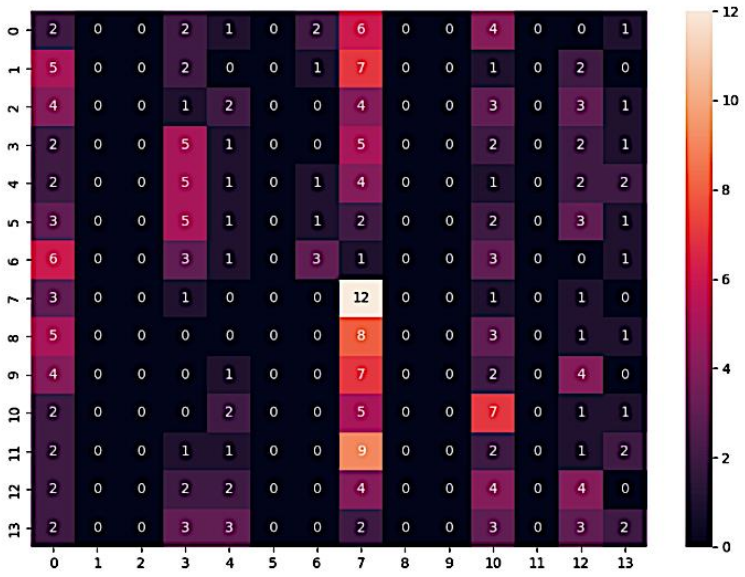

Figure 47 - Classification Heat Map for averaged voxels, last 5 TRs of rest dataset, Vis network

8.5. Analysis of the DMN sub-network

By focusing on sub-networks within the DMN, such as the PFC, pCunPCC, and Temp, we aim to identify specific contributions of these sub-networks to the overall brain activity patterns.
We applied the previous data manipulation techniques (averaging TRs and voxels) to the sub-networks within the DMN. We compared results for:
a. Averaging voxels
b. Averaging TRs

DMN sub-net hierarchy:

network	DMN																	
sub-network	PFC			pCunPCC											Temp			
individual ROI	Default_PFC_1_LH	Default_PFC_2_LH	Default_PFC_3_LH	pCunPCC_1_RH	pCunPCC_1_LH	pCunPCC_2_RH	pCunPCC_2_LH	pCunPCC_3_RH	pCunPCC_3_LH	pCunPCC_4_RH	pCunPCC_4_LH	pCunPCC_5_RH	pCunPCC_5_LH	pCunPCC_6_RH	pCunPCC_6_LH	Default_temp_5_RH	Default_temp_5_LH	Default_temp_6_RH	Default_temp_6_LH	Default_temp_7_LH

Comparison on 3 levels: DMN network, Sub-Networks within the DMN and sample individual ROIs

8.5.1.1. Averaging voxels

	LEVEL						
	Net	**sub-net**			**individual ROIs**		
	DMN	**PFC**	**pCunPCC**	**temp**	**Default_PFC_1_LH**	**pCunPCC_5_RH**	**Default_temp_5_LH**
Eval/Accuracy	0.131	0.095	0.111	0.155	0.071	0.103	0.095
Eval/Loss	2.611	2.616	2.627	2.563	2.631	2.618	2.629
Test/Accuracy	0.115	0.075	0.119	0.147	0.119	0.083	0.135
Test/Loss	2.61	2.63	2.615	2.599	2.629	2.633	2.604
Train/Accuracy	0.137	0.109	0.141	0.153	0.087	0.105	0.133
Train/Loss	2.589	2.614	2.603	2.631	2.609	2.623	2.575

8.5.1.2.Averaging TRs

	LEVEL						
	Net	**sub-net**			**individual ROIs**		
	DMN	**PFC**	**pCunPCC**	**temp**	**Default_PFC_1_LH**	**pCunPCC_5_RH**	**Default_temp_5_LH**
Eval/Accuracy	0.143	0.095	0.115	0.119	0.071	0.091	0.119
Eval/Loss	2.611	2.642	2.589	2.618	2.65	2.652	2.663
Test/Accuracy	0.131	0.091	0.123	0.099	0.071	0.075	0.107
Test/Loss	2.597	2.633	2.616	2.623	2.64	2.636	2.624
Train/Accuracy	0.174	0.11	0.145	0.136	0.072	0.09	0.123
Train/Loss	2.6	2.615	2.59	2.607	2.622	2.608	2.603

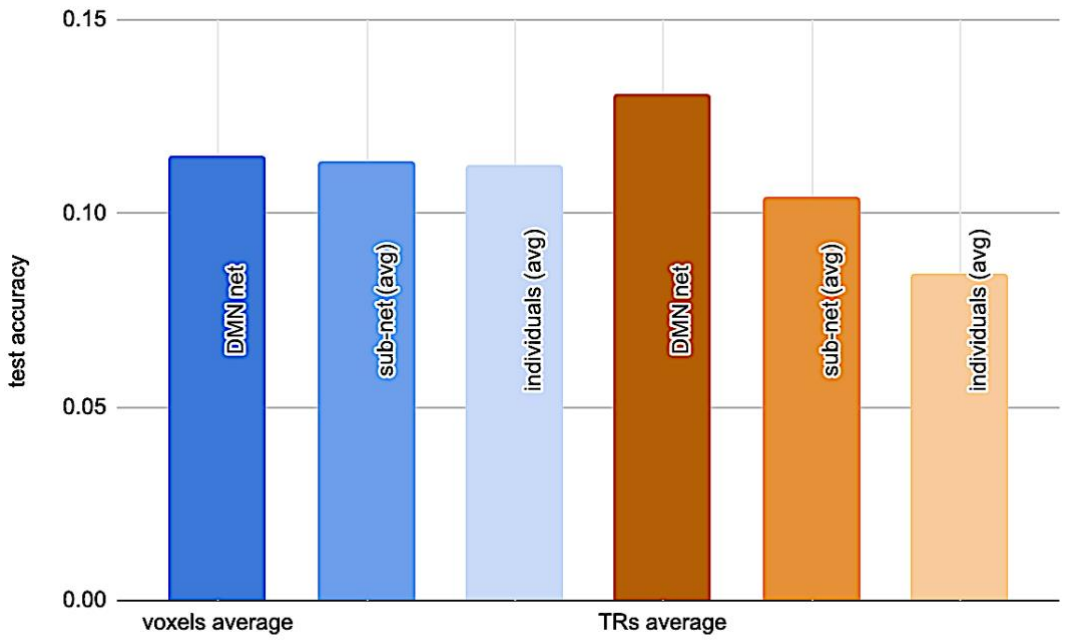

Figure 48 – Comparison between voxels average and TRs average in different levels (network, sub network and individual ROI) for the last 5 TRs of rest dataset.

Summary results:

		test accuracy
voxels average	**DMN net**	0.115
	sub-net (avg)	0.114
	individuals (avg)	0.112
TRs average	**DMN net**	0.131
	sub-net (avg)	0.104
	individuals (avg)	0.084

8.6. Group subjects and cross-validation

Cross-validation
a k-fold cross-validation approach is applied using a consolidated dataset of 176 subjects, organized into cross_val/movies and cross_val/rest folders. The dataset is split into 4 folds, with each fold serving as the test set once, while the remaining three folds are used for training. averaging subjects applied independently within each fold to avoid data leakage. This process is repeated for each fold, and the performance metrics are averaged to obtain the final results.

Workflow:
We implemented a k-fold cross-validation strategy using a dataset of 176 subjects, stored in the `cross_val/movies` and `cross_val/rest` directories. The dataset was divided into 4 folds, each containing about 44 subjects. The choice of 4 folds was determined to be the most effective, balancing the need for a robust test set while ensuring the stability of the performance metrics.
The workflow was as follows:

1. **Separation into Folds**
   The entire dataset was split into 4 folds, ensuring each fold had a roughly equal number of subjects. Each fold served as the test set once, while the remaining three folds were used for training.
2. **Grouping Subjects:**
   Within each test fold, subjects were grouped into sets of 4, by averaging the BOLD signals. This grouping resulted in 11 groups per test fold, with each group treated as a single sample during testing. The grouping allowed us to evaluate the model's performance on aggregated subject data, which provided a more comprehensive assessment.
3. **Testing on Subject Groups:**
   The grouped subjects in each test fold were used to evaluate the model. By averaging the performance metrics across all folds, we obtained final results that reflect the model's generalizability and effectiveness.

hyperparameters	
batch size	8
epochs	10
number of heads	2
dropout	0.7
weight decay	0.01
embedding dim	512
learning rate	0.00005
group size	4
k	4
synthetic	No

results

Vis-5-RH	last 5 TRs of rest	last 5 TRs of movies
averaged test acc	0.271	0.671
averaged train acc	0.341	0.697

Vis network	last 5 TRs of rest	last 5 TRs of movies
averaged test acc	0.494	0.912
averaged train acc	0.882	0.996

DMN network	last 5 TRs of rest	last 5 TRs of movies
averaged test acc	0.453	0.855
averaged train acc	0.966	1

DAN network	last 5 TRs of rest	last 5 TRs of movies
averaged test acc	0.356	0.886
averaged train acc	0.542	0.776

Graphic results

Vis network results:

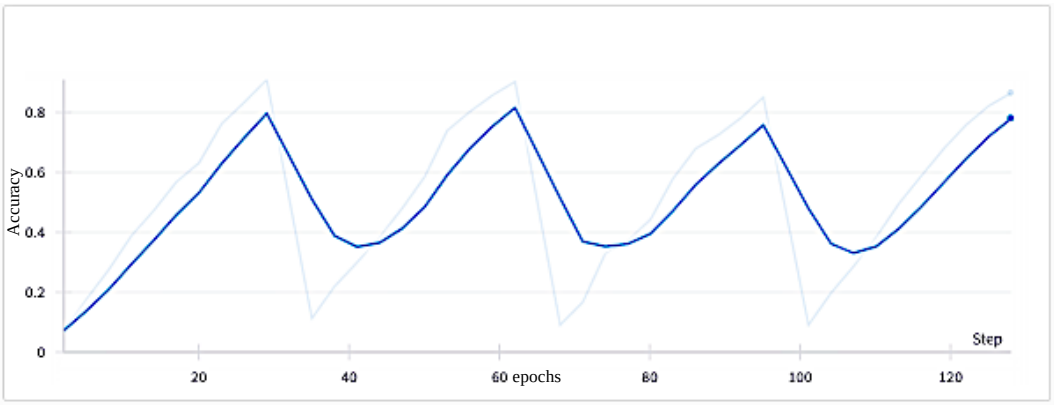

Figure 49 – train accuracy curve – showing train 4 folds of cross-validation and group subjects – Vis network, last 5 TRs of rest dataset.

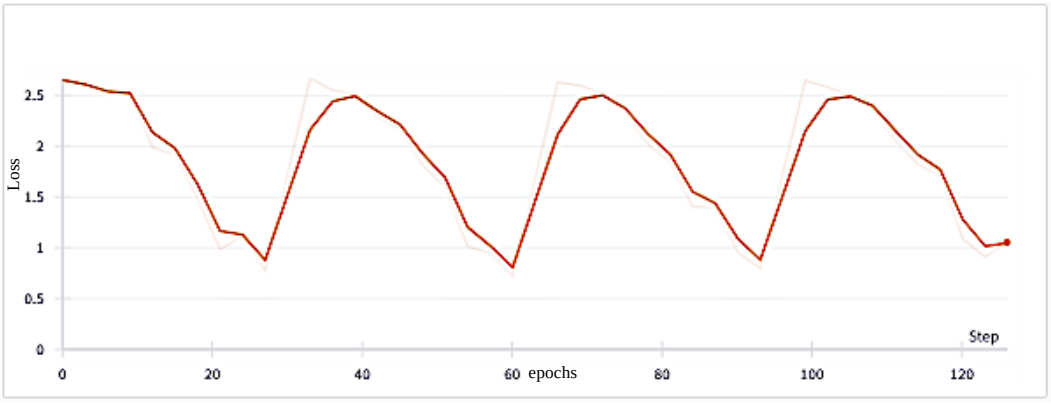

Figure 50 – train loss curve – showing train 4 folds of cross-validation and group subjects – Vis network, last 5 TRs of rest dataset.

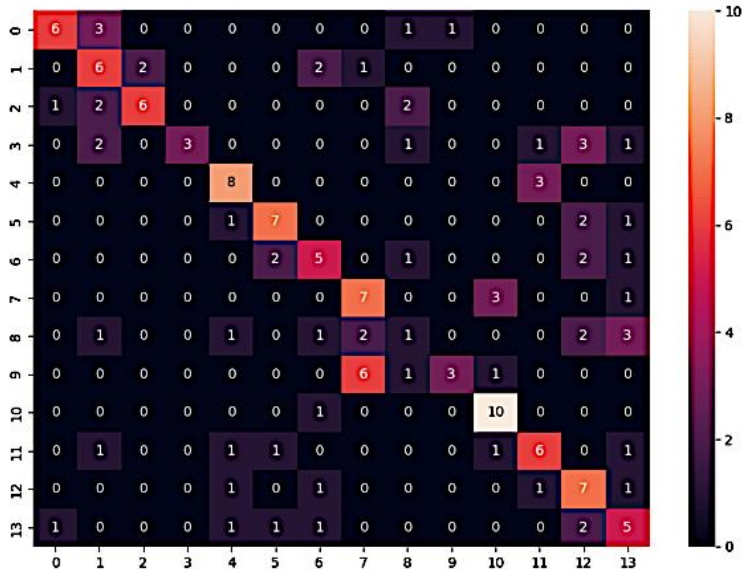

Figure 51 – test classification matrix – cross-validation and group subjects – Vis network, last 5 TRs of rest dataset.

Vis-5-RH results

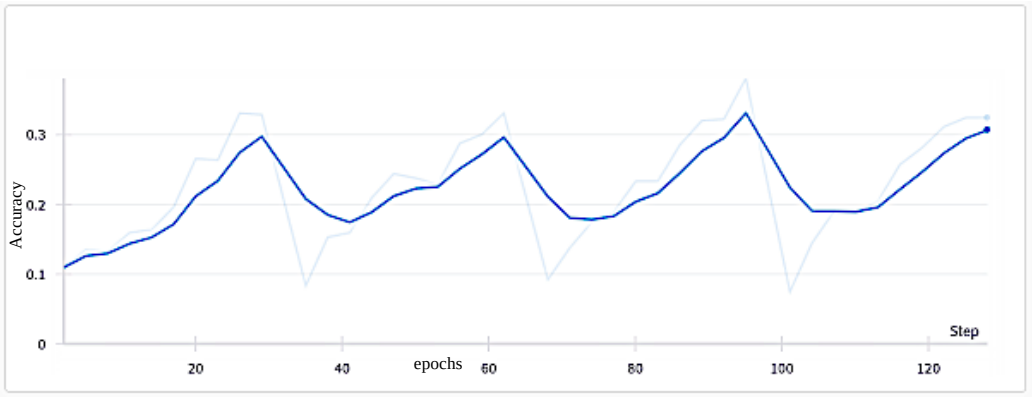

Figure 52 – train accuracy curve – showing all 4 folds of cross-validation and group subjects – Vis-5-RH, last 5 TRs of rest dataset.

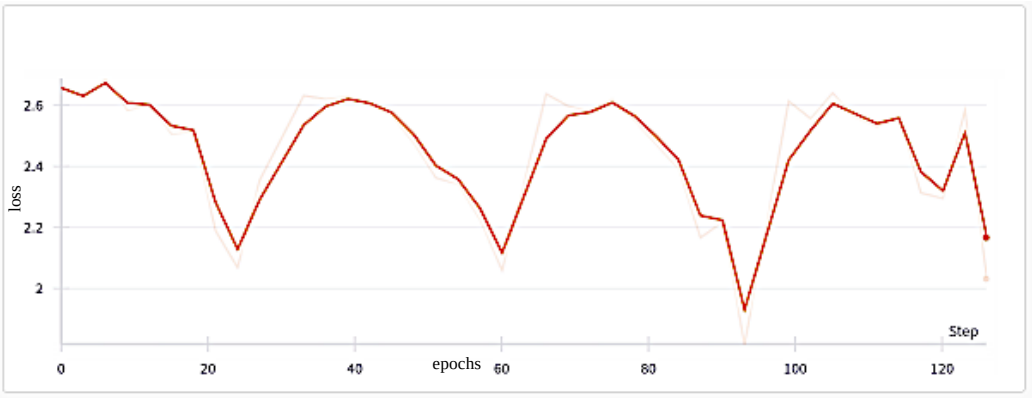

Figure 53 – train loss curve – showing all 4 folds of cross-validation and group subjects – Vis-5-RH, last 5 TRs of rest dataset.

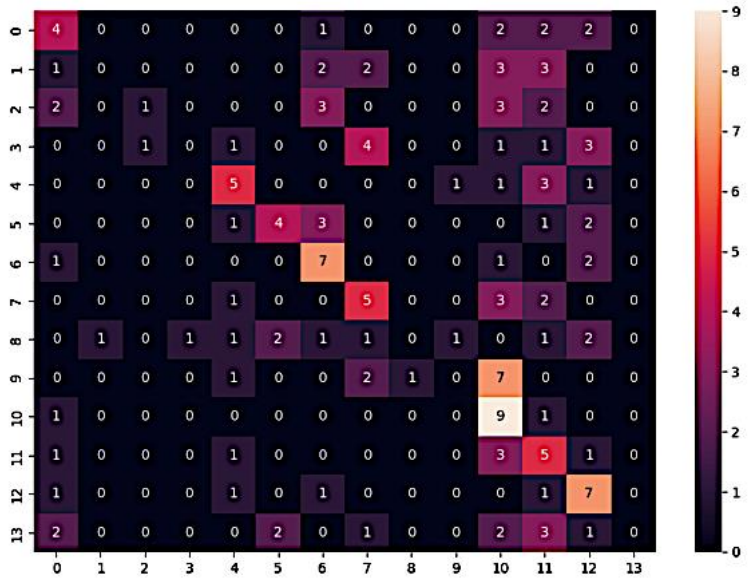

Figure 54 – test classification matrix – cross-validation and group subjects – Vis-5-RH, last 5 TRs of rest dataset.

9. Conclusions

Movies Compared to Rest Between Movies Analysis:

Tuning hyperparameters resulted in noticeable improvements in model performance. For example, with four attention heads and a higher dropout rate of 0.5, the model's performance across all TRs and specific subsets improved in certain ROIs, highlighting the importance of fine-tuning for optimal results. Post-tuning, the model demonstrated high accuracy, confirming its robustness in capturing neural activation patterns throughout the entire movie duration (all TRs).
Despite these improvements, a significant drop in accuracy was observed in the last 15 TRs of the movies dataset compared to all TRs. This drop suggests the increasing complexity of neural activation patterns towards the end of the movies, where the narrative's final segments may elicit more variable brain responses.

End of the Movie:

The model maintained stable performance with an evaluation accuracy of 56% and a test accuracy of 59% in the last 5 TRs. This indicates effective segmentation and classification in the Vis-5-RH ROI, demonstrating the model's robustness despite the challenges inherent in the dataset.

Start of Rest:

A more significant-than-expected drop in performance was observed, with an evaluation accuracy of 53% and a test accuracy of 45% in the first 5 TRs of the rest period. This drop reflects the transitional neural activity from the movie to the resting state, capturing an initial adjustment phase in neural responses.

End of Rest:

A substantial drop in accuracy was recorded, with an evaluation accuracy of 16% and a test accuracy of 20% in the last 5 TRs. This significant decline indicates increased complexity and unpredictability in neural activity during rest, posing challenges for accurate segmentation and classification.

Observations:

The results underscore the model's capability to capture neural activation patterns and memory processes associated with movie-watching. While performance varied across different phases and ROIs, the overall accuracy and insights gained from hyperparameter tuning highlight the potential of transformer-based models in fMRI data analysis. The model also showed promise in capturing memory traces from the rest periods between movie segments.
However, averaging methods did not yield improvements; instead, they highlighted a greater discrepancy between movie and rest periods, with the accuracy difference increasing from 13.5% for the last 5 TRs of the movie to 15.7% for the last 5 TRs of the rest period. This suggests that averaging may not be effective in enhancing model performance during rest periods, highlighting the complexity of neural processes during these intervals.

Specific ROI Observations:

The model's performance dropped significantly after the movie ended. The accuracy at the end of the movie, the start of the rest period, and the end of the rest period were 0.579, 0.472, and 0.202, respectively. The substantial decrease in accuracy towards the end of the rest period may indicate that brain processes during rest are less predictable or less understood by the model.

Averaging TRs:

In an effort to improve accuracy during the rest period, we averaged the TRs. However, this approach led to a further decrease in accuracy to 0.095 at the end of the rest period. This reduction might be attributed to overfitting, as averaging TRs effectively reduces the number of samples (e.g., five TRs condensed into one averaged TR). To address this, we applied PCA and data augmentation methods by creating synthetic subjects. Despite these efforts, the results showed only marginal improvement, with PCA actually decreasing the accuracy to 0.075. Augmentation slightly improved the accuracy to 0.218, close to the baseline without any manipulations.

Data Augmentation of Subjects:

While the introduction of synthetic subjects did not significantly reduce overfitting (and actually increased it), it did lead to a noticeable improvement in test accuracy. This suggests that the model benefited from the additional training data, enabling it to better capture general patterns in the fMRI data across different subjects.
Overall, the use of synthetic subjects with incrementally increasing noise levels proved to be an effective technique for enhancing the model's performance, demonstrating that targeted data augmentation can positively impact complex domains like fMRI data analysis.

Voxel Averaging:

We also attempted voxel averaging instead of TRs. This method did not yield better results, achieving an accuracy of only 0.083 at the end of the rest period.

Combining Visual Areas:

Another strategy involved concatenating all the Vis areas to form a Vis network ROI, effectively combining all individual ROIs within the Vis network. The best result was achieved with this method, particularly when paired with subject augmentation, reaching an accuracy of 0.302 (which is already much higher than the expected accuracy for random data which stands at 0.071).

Analysis of the DMN Sub-Network:

Given that the network-based approach improved accuracy, we conducted experiments comparing individual ROIs, sub-networks, and full networks while testing both voxel and TR averaging. As shown in the results of experiment 8.5.1, none of these approaches significantly outperformed previous methods.

Averaging Across Subjects:

We explored averaging BOLD data across subjects, with every four subjects averaged into one synthetic subject. Due to the limited amount of data, we utilized cross-validation with four folds (k=4). This approach yielded the best results. For the Vis-5-RH ROI at the last five TRs of rest, we achieved an accuracy of 0.271, slightly higher than the original results. For the Vis network, this method resulted in an accuracy of 0.494, the highest observed for the end of the rest period, while the DMN scored 0.453 and the DAN - 0.356.

Averaging across subjects emerged as the most effective method, outperforming both TR and voxel averaging. This could be due to the fact that averaging subjects enhances the signal-to-noise ratio by reducing individual variability, allowing the model to better capture common patterns across subjects. In contrast, averaging TRs or voxels may obscure important temporal or spatial information, respectively, leading to poorer performance.

Conclusion:

This study demonstrated the effectiveness of Transformer-based models in analyzing fMRI data during movie-watching and resting-state periods in between these movies. By optimizing hyperparameters, combining ROIs, and employing subject averaging, the model achieved notable accuracy in classification to 14 different movies. While the model performed well during active movie-watching, it also demonstrated a strong ability to classify movies during rest periods, achieving a high accuracy of 49% when analyzing the last 5 TRs of the rest period. This success highlights the model's robustness in capturing the complex dynamics of brain activity during transitions between active engagement and rest. Averaging across subjects proved to be the most effective strategy, suggesting that combining data from multiple individuals enhances the model's ability to capture common neural patterns.

Our project also suggests that short-term memory of naturalistic movies tends to be more distributed than localized in the human brain.

Future Directions:

1. **Analysis of Transformer's Head Representations:** Further analyze the transformer's head representations to understand which aspects of the rest TRs were accurately recognized by the model and which were not. By probing the attention mechanisms within the transformer, we aim to uncover the specific features and temporal dynamics that the model relies on when processing rest periods.
2. **Encoder-Decoder Transformer Training:** Explore training an encoder-decoder transformer to transition between movie sequences and the memory representations that follow during rest periods.
3. **Transfer Learning:** Contrary to our expectations, training on movie data and then testing on rest data, yielded slightly worse results compared to our standard training and testing procedure. This should be investigated further as we believe the rest testing can benefit somehow from features learnt from the movies data.

10. Glossary of Terms

Brain-Related Terms

Term	Definition
BOLD (Blood-Oxygen-Level Dependent)	A measure used in fMRI to infer brain activity by detecting changes in blood oxygenation levels.
fMRI (Functional Magnetic Resonance Imaging)	A technique that measures brain activity by detecting changes in blood flow, providing insight into the brain's function and structure.
Resting-State fMRI (rfMRI)	A type of fMRI scan where the subject is not performing any specific task, often used to study the brain's functional connectivity.
Repetition Time (TR)	The time between successive pulse sequences in fMRI that determines the temporal resolution of the scan.
ROI (Region of Interest)	Specific areas of the brain that are selected for analysis in neuroimaging studies.
Voxel	A volume element representing a value in 3D space, similar to a pixel in 2D images, commonly used in fMRI to represent brain activity in a specific location.

Machine Learning Terms

Term	Definition
Attention Mechanism	A method in neural networks that allows the model to focus on specific parts of the input sequence, enhancing its ability to capture dependencies and relationships.
Classification	A machine learning task that involves predicting the category or class label of a given data point based on its features.

Confusion Matrix	A table used to describe the performance of a classification model, showing the true positives, false positives, true negatives, and false negatives.
Cross-Entropy Loss	A loss function commonly used in classification tasks to measure the difference between the predicted probability distribution and the true distribution.
Cross-Validation (CV)	A technique for assessing the model's performance by dividing the dataset into K folds and validating the model on each fold in turn.
Decoder	A component in sequence-to-sequence models, such as transformers, that generates the output sequence based on the encoded input and attention mechanism.
Dropout	A regularization technique used to prevent overfitting by randomly setting a fraction of the input units to zero during training.
Encoder	A component in a transformer model that processes the input data and creates a contextual representation that the decoder uses to generate the output.
Epoch	A full pass through the entire training dataset during the training process of a neural network.
Fine Tuning	The process of continuing training a pre-trained model on a new, usually smaller dataset to adapt it to the specific task at hand.
Hidden Layer	Intermediate layers in a neural network that apply transformations to input data before passing it to the output layer, enabling the network to learn complex features.
Hyperparameters	Settings or configurations for a machine learning algorithm, such as learning rate, batch size, and number of layers, that are set before training and not learned from data.
Layer Normalization	A technique used to normalize the inputs of a neural network layer to stabilize and accelerate training.
Learning Rate	A hyperparameter that controls the step size at each iteration while moving toward a minimum of the loss function.

Loss Function	A function that quantifies the difference between the predicted and actual values, guiding the optimization process in neural networks.
Multi-Head Attention	A component of the transformer model that allows the model to focus on different parts of the input sequence simultaneously, improving its ability to capture various relationships.
Overfitting	A scenario where a model learns to perform well on training data but fails to generalize to new, unseen data.
Positional Encoding	A technique in transformers that adds information about the order of the input tokens to the model, allowing it to consider the sequence order.
Preprocessing	The process of transforming raw data into a format suitable for analysis, including steps such as normalization, data cleaning, and feature extraction.
Pretraining	The initial phase in transfer learning where a model is trained on a large dataset to learn general features that can be transferred to a specific task.
ReLU (Rectified Linear Unit)	A widely-used activation function in neural networks that introduces non-linearity by outputting the input directly if it is positive, otherwise, it outputs zero.
Transfer Learning	A machine learning technique where a model pre-trained on a large dataset is fine-tuned on a smaller, task-specific dataset, leveraging the learned features.
Training Set	The subset of the dataset used to train the model by adjusting its parameters to minimize the loss function.
Validation Set	A subset of the dataset used during training to tune hyperparameters and assess the model's performance, helping to prevent overfitting.
Weights	Parameters within a neural network that are learned during training and used to transform inputs as they pass through the network.

11. References

Articles:

[1] Mar, R. A. (2011). The neural bases of social cognition and story comprehension. *Annual Review of Psychology*, 62, 103-134.

[2] Zwaan, R. A., & Radvansky, G. A. (1998). Situation models in language comprehension and memory. *Psychological Bulletin*, 123(2), 162-185.

[3] Schaefer, A., Nils, F., Sanchez, X., & Philippot, P. (2015). Assessing the effectiveness of a large database of emotion-eliciting films: A new tool for emotion researchers. *Cognition and Emotion*, 29(3), 537-555.

[4] Zacks, J. M., Speer, N. K., Swallow, K. M., Braver, T. S., & Reynolds, J. R. (2007). Event perception: A mind-brain perspective. *Psychological Bulletin*, 133(2), 273-293.

[5] Hasson, U., Nir, Y., Levy, I., Fuhrmann, G., & Malach, R. (2004). Intersubject synchronization of cortical activity during natural vision. *Science*, 303(5664), 1634-1640.

[6] Kauppi, J. P., Jääskeläinen, I. P., Sams, M., & Tohka, J. (2010). Inter-subject correlation of brain hemodynamic responses during watching a movie: localization in space and frequency. *Frontiers in neuroinformatics*, 4, 669.

[7] Xu, J., Kemeny, S., Park, G., Frattali, C., & Braun, A. (2005). Language in context: emergent features of word, sentence, and narrative comprehension. *NeuroImage*, 25(3), 1002-1015.

[8] Baldassano, C., Chen, J., Zadbood, A., Pillow, J. W., Hasson, U., & Norman, K. A. (2017). Discovering event structure in continuous narrative perception and memory. *Neuron*, 95(3), 709-721.

[9] Tompary, A., Duncan, K., & Davachi, L. (2015). Consolidation of associative and item memory is related to post-encoding functional connectivity between the ventral tegmental area and different medial temporal lobe subregions during an unrelated task. *Journal of Neuroscience*, 35(19), 7326-7331.

[10] Schapiro, A. C., Turk-Browne, N. B., Norman, K. A., & Botvinick, M. M. (2016). Statistical learning of temporal community structure in the hippocampus. *Hippocampus*, 26(1), 3-8.

[11] Simony, E., Honey, C. J., Chen, J., Lositsky, O., Yeshurun, Y., Wiesel, A., ... & Hasson, U. (2016). Dynamic reconfiguration of the default mode network during narrative comprehension. *Nature communications*, 7(1), 1-11.

[12] Hasson, U., Chen, J., & Honey, C. J. (2015). Hierarchical process memory: memory as an integral component of information processing. *Trends in Cognitive Sciences*, 19, 304-313.

[13] Goldstein, A., Zada, Z., Buchnik, E., Schain, M., Price, A., Aubrey, B., ... Hasson, U. (2022). Shared computational principles for language processing in humans and deep language models. *Nature Neuroscience*, 25, 369-380.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

[15] Caucheteux, C., & King, J. R. (2022). Brains and algorithms partially converge in natural language processing. *Communications Biology*, 5(1), 134.

[16] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence, 35(8), 1798-1828.*

[17] Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., & Eickhoff, C. (2021). A Transformer-based Framework for Multivariate Time Series Representation Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore).

[18] Malkiel, I., Rosenman, G., Wolf, L., & Hendler, T. (2022, December). Self-supervised transformers for fmri representation. In *International Conference on Medical Imaging with Deep Learning* (pp. 895-913). PMLR.

[19] Buxton, R. B. (2002). *Introduction to functional magnetic resonance imaging: Principles and techniques* (pp. 400-424). Cambridge University Press.

Web:

[20] https://en.wikipedia.org/wiki/Lobes_of_the_brain
[21] https://leanpub.com/principlesoffmri/read
[22]https://github.com/ThomasYeoLab/CBIG/blob/master/stable_projects/brain_parcellation/Schaefer2018_LocalGlobal/readme_figures/Schaefer2018_400parcel_parcellation_match_Yeo_7_network_fslr32k.png
[23]https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc
[24] https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb
[25] https://jalammar.github.io/illustrated-transformer/
[26]https://www.humanconnectome.org/storage/app/media/documentation/s1200/HCP_S1200_Release_Reference_Manual.pdf

Link to our project code files on GitHub:
https://github.com/karinsha/fMRI_final_project.git