Utrecht University

# The classification of cognitive operations

by

Rick den Otter

Experimental Psychology

Master Artificial Intelligence, Utrecht University

Supervised by:

Dr Leendert van Maanen

Dr. Surya Gayet

February 16, 2024

## Abstract

This paper explores the application of machine learning techniques to classify cognitive processing operations using EEG data, contributing to the fields of cognitive neuroscience and machine learning (ML). The research discussed is rooted in the theory of processing operations, examining how the brain manages tasks in sequence.

Our methods involve the use of different machine learning algorithms, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs) and transformer networks, to analyze complex patterns in electroencephalography (EEG) data. Our methods are supported by a flexible framework for data collection, processing, and model training, enabling adaptability and integration of new insights. The goal of this research is to develop a proof-of-concept for finding similar cognitive processing operations across different contexts, using the supposed onsets of these operations as provided by hidden multivariate pattern (HMP) analysis (Weindel et al., 2024).

We showcase parameter tests aimed at identifying effective strategies for applying machine learning to EEG data analysis. The results demonstrate the utility of machine learning in decoding brain functioning and answer some questions around which techniques to use in analyzing EEG data within the context of this paper.

The results show that all three previously mentioned classes of machine learning models are able to generalize very well across condition ($\pm 1\%$ loss in performance), slightly worse across lab ($\pm 30\%$ loss in performance), but do not generalize across task. We believe that these results show that generalizing across contexts is principally possible and can function as a first step into discerning what makes each cognitive processing operation unique.

# Table of Contents

## List of Figures

## List of Tables

## 1 Introduction

What goes on in the brain has intrigued science for a very long time, as early as 1868 Donders introduced reaction time (RT)-based experiments (Donders, 1868), laying the groundwork of what is now known as cognitive science. Donders' ideas led to many new insights and paths of research. Since the cognitive revolution in the 1950s (Miller, 2003), the field of cognitive science has focused increasingly on what drives the brain and which internal processes guide cognitive functioning. Advances in neuroimaging techniques, along with statistical and computational methods to process data coming from neuroimaging experiments, have led to greater functional understanding of the brain and cognition.

There are many theoretical proposals that assume cognition can be understood in terms of discrete cognitive operations that unfold sequentially through time. These proposals are all based on Donders' method of subtraction. Suppose an experiment where one variable differs, creating an additional cognitive operation. If we then subtract the control condition reaction times from the condition with the additional cognitive operation, we end up with the time it takes for that added operation to execute.

The method of subtraction is implicitly present in many works, such as Miller's theory on information processing (Miller, 1956), Broadbent's filter model of attention (Broadbent, 1958), Atkinson & Shiffrin's multi-store model (Atkinson & Shiffrin, 1968), Sternberg's method of additive factors (Sternberg, 1998), and Zylberberg's human turing machine (Zylberberg et al., 2011). Indirect evidence for these models and methods has been provided in the form of experiments that validate their predictions. It is also explicitly modelled in cognitive architectures such as Soar (Laird, 2019) and ACT-R (Anderson, 2007), again validated through experiments compared with model predictions.

These approaches all require theorizing about the cognitive operations under consideration. This leaves room for error and may not be desired, for example in the case of individual differences, where different individual strategies may introduce additional operations, or skill learning, where an operation may shorten in duration or disappear as the individual grows more skilled. In this paper, we propose a novel method to directly identify cognitive operations based on their neurophysiological signature. Conceptually, the idea is as follows:

1. Identify onsets of cognitive operations in a task.
2. Learn the neurophysiological signature of a cognitive operation.
3. Identify those signatures in unseen data.

Here, we discuss a proof-of-concept that implements these steps. We include gathering and preparing data, selecting methods from the machine learning literature, and visualizing what a model has learned about the cognitive processing operation. Firstly, we introduce the foundation of this research, determining cognitive processing operations through neuroimaging techniques.

Despite substantial progress in unraveling the mysteries of the brain and cognition through neuroimaging techniques, considerable gaps remain in our understanding of the neural mechanisms underlying complex cognitive processes like learning, recall, and attention. One cognitive framework offering promising insights into this domain is the theory of processing operations. This theory postulates that a series of partially serial operations are the neural foundations of cognitive tasks (Anderson, 2007; Newell, 1994; Zylberberg et al., 2011). Processing operations have been integral to models of information processing like ACT-R (Anderson, 2007) and Soar (Newell, 1994). Neural correlates of processing operations in functional magnetic resonance imaging (fMRI) data have been found in (Anderson et al., 2008; Borst & Anderson, 2013).

Existing research within this field has mostly operated at a temporal resolution of 0.1 seconds, aligning with Newell's cognitive band (Newell, 1994). Studies examining processing operations through neuroimaging data have observed that each operation is characterized by a distinct peak of neural activity, or an 'event' (Anderson et al., 2016; Borst & Anderson, 2015, 2021).

Despite extensive research in separate domains—cognitive processing operations, machine learning, and neuroimaging data analysis—there has been an absence of studies merging machine learning with cognitive processing operations. We bridge this interdisciplinary gap by addressing the central research question:

> To what extent can machine learning models accurately classify cognitive processing operations, and how generalizable is such a classifier across different contexts or populations?

Through the integration of machine learning techniques with the framework of cognitive processing operations, this paper contributes to the ongoing effort to unravel the complex interplay of neural and cognitive processes.

# 2 Background

The exploration of the human mind has always been of great curiosity and interest to the scientific community. Cognitive science offers a unique understanding of mental processes and the neural mechanisms underlying them. Central to cognitive science is studying how information is processed by the brain and how these processes can be understood and modelled through computational models. This paper dives into two branches of cognitive science: models of information processing, and cognitive processing operations. These domains are interconnected and together provide a comprehensive framework for examining the topics that are relevant to our paper. Within this context, we will also explore four other topics essential to the research question: preprocessing of EEG data for machine learning (ML), ML applied to neuroimaging data, generalization in ML for EEG data, and feature visualization.

## 2.1 Models of information processing

The exploration of models of information processing in this section supports viewing cognition as an information processing system. The models that will be discussed transform this theory to a framework that allows for deconstruction and analysis of cognitive processes. By modelling cognition as an information processing system, we gain valuable insights into how the brain processes information. This understanding is vital for our research since it provides a foundation for interpreting and analyzing EEG data.

Models of information processing also shed light on the mechanisms underlying cognitive operations. This is important when investigating more complex cognitive tasks, where multiple processing operations are involved. Understanding these stages helps with developing accurate machine learning models that can classify and predict cognitive operations from EEG data. In summary, the study of information processing models not only enhances our theoretical understanding of cognitive science but provides practical capabilities in applying techniques like machine learning to neuroimaging data.

We will discuss several models of information processing that introduce foundational ideas leading to the theory of cognitive processing operations.

Explicitly identifying processing operations in models of information processing started with Broadbent in 1958, who introduced the filter model of attention (Broadbent, 1958). He proposed that information processing involves a series of stages, including an early ‘filter’ stage that allows only certain information to pass, based on physical characteristics. He theorized that the filter stage is followed by stages that employ a more in-depth processing approach.

In 1968, Atkinson & Shiffrin developed the Atkinson-Shiffrin, or multi-store model which proposed a series of stages that information passes through in memory, including sensory, short-term, and long-term memory (Atkinson & Shiffrin, 1968). The sensory memory is where information enters through the senses, holding information for short periods of time. Short-term memory holds small amounts (seven plus or minus two (Miller, 1956)) of information, and was at that time assumed to contain conscious thought processing. Information that is in the short-term memory can either be forgotten or saved to long-term memory. Finally, long-term memory can be seen as a permanent storage system with essentially unlimited storage capacity. Information in long-term memory can be brought back into short-term memory through the process of retrieval.

Following up on the multi-store model, Craik & Lockhart (F. I. M. Craik & Lockhart, 1972) and Baddeley & Hitch (Baddeley & Hitch, 1974) proposed alternative models of information processing. Craik & Lockhart look more closely at memory. They define three tiers of mental processing, increasing in depth. The depth affects how well information is remembered. Baddeley & Hitch focus on the 'central executive', which acts like a manager of attention, directing it to particular tasks depending on where attention is needed.

Skipping a few years ahead, we arrive at the concept of Parallel Distributed Processing (PDP). This idea was introduced by McClelland, Rumelhart, and the PDP research group in 1987 (McClelland et al., 1987). PDP is the first model that suggests that cognitive processes occur in parallel and are distributed across a network of interconnected units. This idea heavily influenced the field of AI, specifically machine learning, and started a major shift of research towards connectionism. Learning in PDP models occurs through changes in the strengths of connections between processing units, or weights. PDP employs Hebbian learning, which involves adjusting weights based on the correlation of activation between nearby connected units. Complex behavior emerges through the interaction of many simple units and knowledge can be represented as patterns of activation over the units.

## 2.2 Detecting cognitive processing operations in neuroimaging data

Reaction time (RT)-based experiments have since as early as the 19th century (Donders, 1868) been used as guidelines for modelling human cognition in models of information processing and in cognitive architectures such as Soar (Laird, 2019) and ACT-R (Anderson, 2007). These cognitive architectures are defined and validated based on the correlation of their simulation-times with reaction times of experiments.

But what makes up reaction time? Can we view the cognitive aspect of task performance as atomic, or can it be broken down into smaller parts? This question has interested cognitive scientists for decades. Many researchers (Anderson et al., 2016; Borst & Anderson, 2015, 2021; Sternberg, 1998; Zylberberg et al., 2011) have argued in favor of cognitive processing operations as the basic unit making up task performance.

Cognitive processing operations serve as the brain's modular approach to handling tasks and processing information. In cognitive processing, the brain is seen as processing information in a sequence of steps. This theoretical framework offers a more nuanced understanding of cognitive functions.

Depending on the specific task, the composition of these operations can vary significantly. For instance, during the initial 'input' stage, there can be a wide array of sensory inputs, whether auditory, olfactory, or otherwise. As the brain advances to the 'processing' stage, the procedures it employs can differ depending on the intended use of the incoming information. Lastly, the 'output' stages are dictated by the ultimate action being taken—be it information storage in memory systems or the initiation of a motor response.

In the last decade, research has been done on finding processing operations in EEG data gathered on humans performing experimental tasks. To illustrate, in (Anderson et al., 2016; Borst & Anderson, 2015, 2021), Borst and Anderson employ hidden semi-Markov models (HsMM) (Yu, 2010) combined with multivariate pattern analysis (MVPA) on EEG data. These statistical ML models are able to model data as a sequence of discrete states with variable duration. Borst and Anderson use the HsMM-MVPA method to research the decomposition of reaction time in a specific task into the processing operations that it contains. The HsMM-MVPA method is able to locate sinusoidal peaks (bumps, or events). These events are theorized to occur at the onset of every cognitive processing operation.

Since 2015, this method has improved and been successfully used to make convincing arguments on which processing operations occur in cognitive experiments. For example, in (Berberyan et al., 2020, 2021; Borst & Anderson, 2021; van Maanen et al., 2021) the method was used as (part of) the methodology. HsMM-MVPA is used in these papers for validating the method's performance or for serving as a baseline of the number and length of processing operations which occur in the experiment.

To apply the HsMM-MVPA method to EEG data, the Python package HsMM-MVPy (HMP) (Weindel et al., 2024) has been developed. HMP offers varying functionality for analyzing neuroimaging data using the HsMM-MVPA method, allowing for customization of parameters and visualization of results.

Using the discussed theory of HsMM-MVPA, we are able to locate the onsets of processing operations with some confidence and theorize about their purpose. With this information we can attempt to increase certainty about the type of processing operations in EEG data using ML.

## 2.3 Preprocessing of EEG data for machine learning

An electroencephalogram (EEG), is a noninvasive procedure that records electrical patterns in the brain (Schomer & Lopes da Silva, 2017). It is an important tool in the field of neuroscience, offering valuable insights into many aspects of cognition, including cognitive processing operations and neurological disorders.

The brain is composed of billions of neurons, which communicate through electrical signals. When groups of neurons send these signals simultaneously, the activity generates electrical fields that can be detected from outside the skull. EEG measures these fields to create a record of the electrical activity occurring in the brain over time.

To gather EEG data, a set of electrodes is attached to the scalp. These electrodes are typically placed in a standardized arrangement to ensure consistent data collection across individuals and studies. They measure the voltage fluctuations resulting from ionic current flows within the neurons of the brain. This information is then transmitted to a computer, where it is graphically represented as waveforms.

The way EEG data is preprocessed is important to consider, since slight variance in electrode position or external factors can cause EEG data to be skewed or even unusable. Since electrodes are placed on the scalp, and not on the brain itself, only about 5% of the actual brain signal is recorded. In preprocessing of EEG data, three major steps can be identified (Cohen, 2014). Artifact removal, which attempts to remove uninformative information from the data, like eye-blinks or muscle spasms. Frequency band selection, for most applications, only a part of the recorded frequencies is relevant. A frequency band can be selected by applying low-pass and/or high-pass filters. Finally, EEG channel selection can be used in situations where not all channels or electrodes are informative for the task.

From (Altaheri et al., 2021; A. Craik et al., 2019), it can be seen that about two-thirds of studies either explicitly mention leaving artifacts in, or do not address the issue at all. The remaining third is split between manual removal and automatic removal, using techniques such as Independent Component Analysis (ICA) (Delorme & Makeig, 2004).

For frequency band selection, (Altaheri et al., 2021) cites many studies that experimented with different frequency filters. Their conclusion is to use either raw EEG signal, or filtering the EEG signal with a low-pass (below 38 Hz) filter. This conclusion is however made specifically for motor imagery tasks, which, according to the paper, utilize very low frequencies (0.5-5 Hz). In studies examined by (A. Craik et al., 2019) about half of the studies employ a low-pass filter at or below 40 Hz. Both reviews mention a lack of studies explicitly aimed at finding out if the performance of an ML model drops without filtering signal frequencies.

Channel selection is mentioned as an important step in (Altaheri et al., 2021), less so in (A. Craik et al., 2019). This is because for applications in motor imagery, it helps to consider only those channels that correspond to motor imagery activation. The authors even show that one motor imagery-correlated channel is often sufficient for classification. In (A. Craik et al., 2019), channel selection is mostly disregarded because sufficiently complex models seem to be able to learn from multivariate relationships between channels.

## 2.4 Machine learning applied to EEG data

With the large and varied datasets EEG experiments provide, ML seems a prime solution for some problems in cognitive science. This opportunity seems to be recognized by the field, as (deep) ML has been applied to varying problems, such as emotion recognition, motor imagery, mental workload, seizure detection, sleep stage scoring, and event related potential detection (A. Craik et al., 2019).

Varying types of ML have been used on EEG data, Craik et al. (A. Craik et al., 2019) and Altaheri et al. (Altaheri et al., 2021) provide an overview of recent research in their reviews. They show that CNNs take up the largest portion of models used at 43%. When looking at studies in motor imagery, this percentage goes up to 78%. Both reviews mention that this high percentage is partly due to the proven success of CNNs in other ML problems, and partly due to the accessibility ML libraries such as TensorFlow (Abadi et al., 2015) and PyTorch (Paszke et al., 2019) provide for CNNs.

Inspired by the ideas of McClelland and Rumelhart (McClelland et al., 1987), artificial neural networks (ANNs) and specifically CNNs are networks designed to automatically learn spatial hierarchies of features from input data. CNNs have shown to be highly effective for tasks in varying fields, such as computer vision, automated driving, healthcare, and robotics. A CNN consists of convolution layers (see Figure 1), which learn filters to detect certain features in their input. These are often followed by pooling layers, which reduce the dimensionality of the input feature map, retaining the most important information. Deeper CNNs often consist of multiple modules of these layers, with different parameters. After one or several of these modules, fully connected layer(s) are used to classify the input. Besides this, it is important to introduce non-linearity into the model through the activation function. A linear model would only be able to fit to linear relationships in data, while non-linear models can capture more complex relationships.

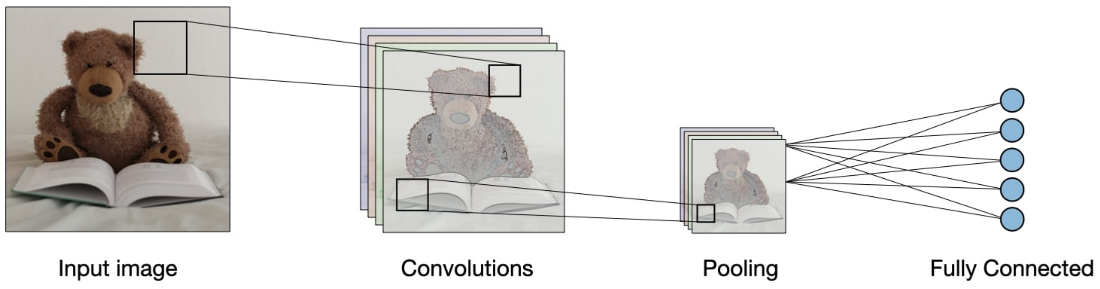

Figure 1: Graphical representation of a convolutional neural network (Afshine Amidi & Shervine Amidi, n.d.)

Besides CNNs, other approaches have also been used successfully. One example is the RNN, which uses recurrent layers that have backward (or residual) connections (see Figure 2). These models are commonly used for extracting time-sensitive information with long-term relationships in the data. They have been successfully used in machine translation (Cho et al., 2014) and language models (Mikolov et al., 2011). A problem with larger models is the vanishing gradient problem (Hochreiter, 1998) where more and more sequential operations cause values and weights in the network to become smaller and smaller over time, making the network less effective. Long short-term memory (LSTM) models (Hochreiter & Schmidhuber, 1997) were created to mitigate the vanishing gradient problem. They can be seen as an extension to RNNs and function by maintaining a sort of memory, capturing information about what has been calculated earlier. This makes LSTMs better at learning long-term relationships. Finally, gated recurrent unit (GRU) models (Cho et al., 2014) were introduced as a computationally more efficient version of the LSTM model. The LSTM and GRU models introduce new ways to decide which information to keep, giving new versions of the hidden (RNN) cell as seen in Figure 2. The LSTM cell adds three gating mechanisms (Hochreiter & Schmidhuber, 1997), which learn to decide which information passes through. The forget gate decides which information to keep from the previous state, the input gate decides which information from a new state to store in the current state, and the output gate considers the previous and current states to decide which information to output. A GRU, in comparison, does not use an output gate, but only uses gating mechanisms to learn which features to input and forget.

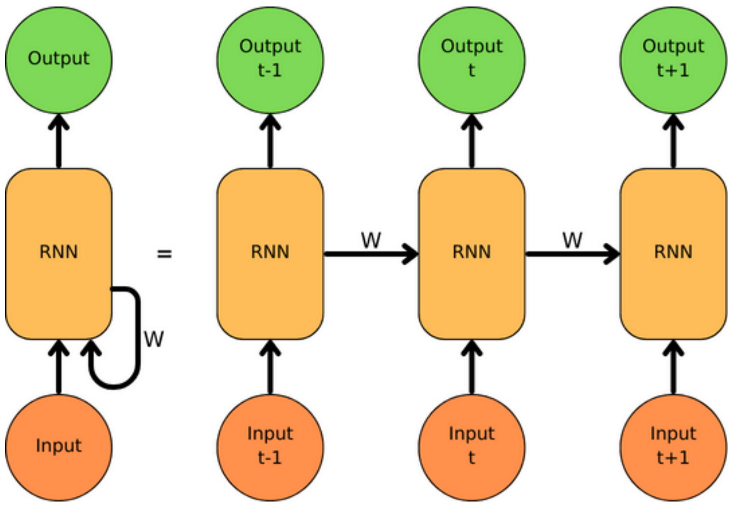

Figure 2: Graphical representation of a recurrent neural network. At each time point the weights in the internal state (marked RNN) are updated, resulting in an output for that point.

RNNs have also been used as part of an encoder-decoder architecture. These are used in sequence-to-sequence prediction problems such as machine translation (Cho et al., 2014) and text summarization (Sutskever et al., 2014). They consist of two models, generally RNNs are used. The encoder processes the input sequence and summarizes the information into its internal state. The encoder's output is a context vector, describing the input sequence. The decoder then uses the context vector to start generating the output sequence. During generation, the decoder is influenced by every previous step. An extension to the encoder-decoder architecture is the concept of attention (Bahdanau et al., 2014). Attention helps the model focus on different parts of the input when generating output. Instead of encoding the input sequence into a single fixed content vector, the attention model develops a context vector that is filtered differently depending on the output step.

Lastly, in (Vaswani et al., 2017), the concept of attention is taken one step further, disregarding recurrence entirely. The transformer model was initially designed for translation tasks but has been widely used, recently popularized as the backbone of large language models such as ChatGPT (Zhao et al., 2023). A transformer model works by embedding the input sequence into a vector representation, combining this representation with a positional encoding vector. The information then goes through a familiar encoder-decoder model with an important distinction. Instead of residual connections, self-attention is used. Self-attention allows each position in the sequence to attend to all other positions. This mechanism is useful for creating an understanding of the context and dependencies of sequence items.

Models with convolutional layers have shown most success for spatially informative data, this is important to consider when formulating input data for such a network. The process is similar to feature selection for other datasets, since in this step, it is decided what information the network has access to. The different types of input formulation that have been used are divisible into four main categories: raw signal values, extracted features, spectral images, and topological mappings. In (Altaheri et al., 2021), it is shown that formulating EEG data into a topological mapping of $time \times height \times width$ outperforms a simpler mapping of $time \times channels$. The height and width dimensions should correspond to the electrode layout.

There has also been research into using deep learning for automatic denoising of EEG data, showing promising initial results (H. Zhang et al., 2021). This could help with some of the more manual steps in preprocessing EEG data.

## 2.5 Generalization in machine learning for EEG data

Most research that uses machine learning for EEG data focuses on generalizing to a test set from the same context (Altaheri et al., 2021), this paper aims to find the model that best generalizes to data from different contexts. Different generalization methods can be employed to improve generalization performance within the same context and to a different context. We will cover some of these methods that are most relevant to EEG data.

Choices that are consequential for generalization performance are made as early as the preprocessing of data. Normalization is important to create a model that learns equally from all numerical features (Santhakumaran, 2011), different techniques can be used depending on the data. The simplest of these is min-max normalization where the minimum and maximum of the data are transformed into a lower bound and upper bound, commonly 0 and 1, respectively. All other values are squashed between the minimum and maximum. Another method that is perhaps more commonly used in EEG research is Z-scoring the data, which measures how many standard deviations are between any data point and the mean of the data and scales the data accordingly.

A common problem in EEG research is a lack of data, which can be partially solved using data augmentation. Data augmentation is often used in for example image recognition to increase the variability and size of the dataset, making the model more robust. In (Lashgari et al., 2020), the authors show that different data augmentation techniques offer performance improvement over the baseline performance, tested on EEG data. Examples include noise addition, which adds some semi-random (often Gaussian) noise to the data, and using a generative adversarial network (GAN) that is taught to generate additional training samples resembling the dataset.

Not all ML models are created equal, deciding to add another layer or increasing the amount of units used in a layer can have a large effect on generalization performance. Careful consideration of the model architecture based on the complexity of the problem is required to ensure that the model captures underlying patterns and relationships correctly. Failure to do so means that the model is unfit to capture these relationships and likely not complex enough. If a model learns the noise and outliers in a training set instead of general features, we call it overfitted. Preventing these failures can be done by for example grid search (LaValle et al., 2004) on model architecture parameters, which finds the best performing configuration by validating performance for each of the parameter combinations it is given.

After choosing the type of model and deciding on its architecture, there are additional techniques that can increase generalization to a test set, we call these techniques 'Regularization'. Defined by (Kukačka et al., 2017) as:

> Any supplementary technique that aims at making the model generalize better, i.e. produce better results on the test set.

We can examine some of these techniques in turn. Perhaps the easiest to understand method of regularization is dropout (Srivastava et al., 2014), which randomly (at a provided probability) drops units from the network during training. This technique essentially simulates an ensemble network by testing subsets of the main network and combining what is learned from these subsets. A second pair of regularization techniques is L1 and L2 regularization, also known as weight decay (Schmidhuber, 2015). These techniques both aim to decrease the weights, without losing information. Both add calculated penalty values to the loss value of the model. L2 regularization punishes larger weights, which are more likely to be abnormal. L1 regularization decreases all weights equally. Lastly, there exist also some in-training normalization techniques, batch (Ioffe & Szegedy, 2015) and layer (Ba et al., 2016) normalization. These normalization techniques solve the problem of a changing distribution of layer inputs as the model learns. Batch normalization sets the mean and variance of layer inputs per-batch. Layer normalization works on a per-sample basis and calculates the mean and variance over all inputs to a layer. The last method of regularization that will be discussed is label smoothing (Müller et al., 2020), this method changes the hard (0, 1) target values to a weighted average of the targets, uniformly distributed across the labels. This prevents the network from over-fitting to a predominant label class.

## 2.6 Feature visualization

Within the context of applying machine learning to EEG data, it is important to look further than performance metrics. Being able to examine the features learned by the model is critical in understanding not only the models, but also the cognitive operations that the model is recognizing. Different types of explanations have been used to make models more interpretable, separable based on the following taxonomy (Molnar, 2022):

- A *model-agnostic* method separates the explanation from the machine learning model, no access to the model's weights is required. These methods can be applied to any model. A *model-specific* method works on one model type only.
- A *global* method describes expected performance averaged over the entire model, a *local* method explains individual predictions.

For neural networks specifically, local model-specific methods are often used since the features a network learns are embedded in its hidden layers and the configuration of its weights. These local methods are also often more computationally efficient since they can calculate the required information directly instead of being based on model predictions. We will cover two methods, since a combination of them is most relevant for feature visualization in EEG machine learning models.

Feature visualization is a method that attempts to reveal information about the features a model has learned. It does so by finding the input that maximally activates a certain unit, which can be a single neuron, feature map, or entire layer.

Pixel attribution is often used in image classifier models that tries to explain the prediction a model makes by attributing importance to each pixel in making that prediction. These methods are often perturbation-based, making small changes to the input and observing how the model reacts to those changes. Examples include SHAP (Lundberg & Lee, 2017) and LIME (Ribeiro et al., 2016).

Specifically for EEG data, feature visualization methods that attribute importance to features can be used. It is important however to consider the type of data that the feature visualization method can be used on. Many methods are meant for images and perturb pixel values. These methods often do not work well with waveform information inherent in EEG data.

A method that works for EEG data is integrated gradients (Sundararajan et al., 2017), this method calculates the gradients from a carefully selected baseline to the data by considering the path in feature space from the baseline to the data point. Features, or electrodes, that contribute more to this path are attributed a higher value.

# 3 Methods

In this section, we describe the methodological framework used to determine whether it is possible to classify cognitive processing operations. Central to our approach is the use of EEG data as the primary source of information. To dissect this multivariate data, we employed different machine learning algorithms, including CNNs, RNNs, and transformer networks. Each offers unique strength in spatial pattern recognition and temporal data analysis. The methods section is organized as follows: First, we detail the procedures for data gathering and preparation. Then, we delve deeper into the specifics of each machine learning model employed, explaining choices in their architecture and the differences that make each model suited to decode patterns in EEG data. Afterwards, we discuss how we compared the performance of the models to baseline classifiers. Then, we consider each of the parameter tests that were performed in order to answer ancillary questions that arose during research. Finally, we examine the methodology applied for testing generalization across contexts.

## 3.1 Data gathering

For this paper, we established a flexible data gathering, processing, and model training pipeline, which can be found at a public GitHub repository (den Otter, 2024). This pipeline was designed to be easily adaptable to changes in input data or model parameters. This adaptability proved essential for incorporating new insights and exploring implications.

The primary dataset for our analysis was collected from an experiment on the speed-accuracy trade-off (SAT1) (Boehm et al., 2014) in perceptual decision-making (see Figure 3, UvA), involving 25 participants. In this task, participants were asked to indicate the direction of motion from a cloud of moving dots. Participants were instructed to focus on speed in half of the trials, and accuracy in the other half. The order of speed and accuracy trials was completely randomized. Earlier work has identified processing operations in this task, which partially overlap across conditions (van Maanen et al., 2021). The cognitive processing operations were labelled as *pre-attentive*, *encoding*, *decision*, *confirmation*, and *response*, see Figure 3 for a visual representation. In this figure, each row represents a different context, whether determined by the condition, lab, or task. Each unique color represents a different cognitive processing operation. The first two rows then represent the SAT1 experiment, containing the theorized sequential nature of stages within the task. Real data would contain stages of variable length.

A preprocessed version of this dataset (Weindel, 2021) was used to fit an HMP model. Preprocessing steps included:

- Bandwidth filtering between 1 and 35 Hz.
- Artifact removal using ICA (Delorme & Makeig, 2004).
- Applying autoreject (Jas et al., 2017) to identify and rectify bad epochs.

Both the preprocessed dataset and a new version of the dataset without manual preprocessing were used in this paper.

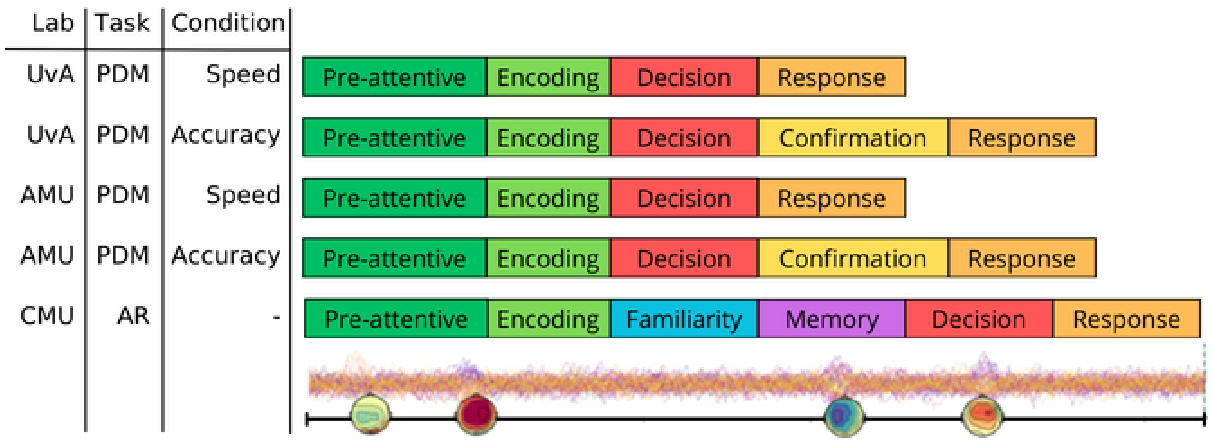
Lab | Task | Condition
UvA | PDM | Speed
Pre-attentive | Encoding | Decision | Response
UvA | PDM | Accuracy
Pre-attentive | Encoding | Decision | Confirmation | Response
AMU | PDM | Speed
Pre-attentive | Encoding | Decision | Response
AMU | PDM | Accuracy
Pre-attentive | Encoding | Decision | Confirmation | Response
CMU | AR | -
Pre-attentive | Encoding | Familiarity | Memory | Decision | Response

Figure 3: Expected processing operations across different labs, task, and conditions visualized. Lengths not representative of reality, EEG data used as visual aid. UvA: *Universiteit van Amsterdam*, AMU: *Aix-Marseille Université*, CMU: *Carnegie Mellon University*, PDM: *Perceptual decision-making*, AR: *Associative recognition*

A second dataset, used to test generalizability over labs, comes from a similar speed-accuracy trade-off experiment (SAT2) (Weindel et al., 2021) (see Figure 3, AMU). This dataset was chosen to test whether the models could generalize to a dataset with the same processing operations, tested at a different lab. The processing operations are expected to be the same as the SAT1 dataset. In this experiment, 20 participants were asked to discern which of two Gabor patches (sine wave grating) had higher contrast. Participants were instructed to focus on either speed or accuracy during the experiment and received feedback on their performance after each trial. The electrodes in this experiment used the same placement, additionally including 32 extra electrodes. For the purposes of this paper, these electrodes were excluded.

Apart from the difference in experiment design (Gabor patches as opposed to random dot motion), the SAT2 dataset also included varying force level modifications on the response button. This was intended to better control for confounding effects of fatigue and learning (Weindel, 2021). Additionally, the SAT2 dataset manipulated the speed-accuracy condition per-block, instead of a randomized order.

The final dataset was used to test generalizability over task contexts and comes from an associative recognition (AR) experiment (Borst et al., 2013). In this experiment, 20 participants were asked to remember shown word pairs during the training phase. In the testing phase, three kinds of word pairs were shown to the participant:

- Targets, where the exact pair was shown during the training phase.
- Re-paired foils, which contain one word that was shown as part of a pair during the training phase and one that was not.
- New foils, which consist of two new words that were not shown during the training phase.

Participants were asked to judge whether word combinations were previously experienced as a pair.

In an earlier analysis (Borst & Anderson, 2021), HMP was applied to detect processing stages. Six were found, labeled *pre-attentive*, *encoding*, *familiarity*, *memory*, *decision*, and *response*. See Figure 3, AR for a visual representation. These labels partly overlap with the labels identified for the SAT tasks, therefore, we predict that a model trained on stages with similar labels in another task will be able to classify these stages above chance.

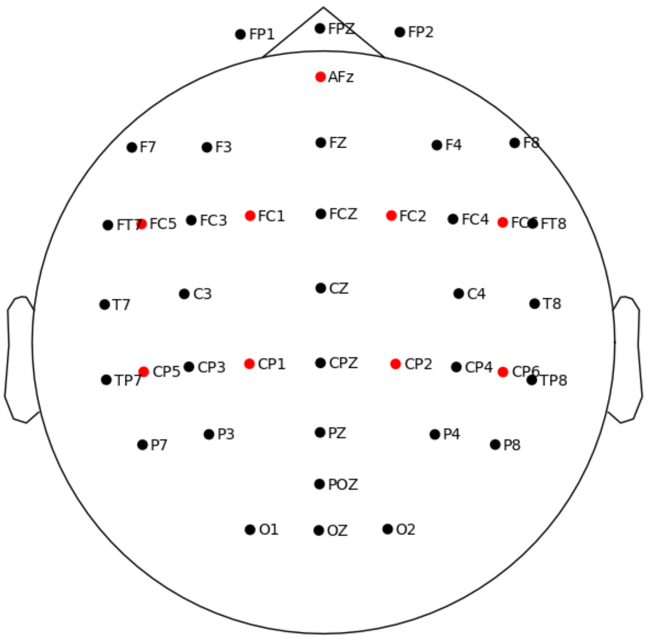

Figure 4: Combined electrodes in SAT and AR experiments, the color red denotes an electrode that occurs in the SAT electrodes but not in the AR electrodes. The reverse did not occur.

Additional steps were necessary to use the data from the associative recognition experiment, as the electrode positions differed slightly from the previous experiments. Investigation showed that each electrode included in the SAT experiments but not in the AR experiment had close electrodes on both sides, meaning that interpolating these signals was feasible. The AR data was also mastoid-referenced instead of average-referenced. We interpolated the missing electrodes and removed electrodes that were not present in the AR experiment (see Figure 4) using the MNE-Python software package (Gramfort et al., 2013). We also average-referenced the AR data using MNE.

### 3.1.1 Processing

The preprocessed data was used to fit an HMP model (Weindel et al., 2024). We used the `fit_single()` method to fit the model since the number of events was known beforehand. The number of starting points was set to 100 to have a higher chance of finding good event timings. Default parameters were used for tolerance (0.0001), and max iterations (1000). The width of the half-sine event to be detected in EEG data was set to 50 milliseconds, and the shape of the gamma distribution to 2. The fitted HMP model gives us trial-by-trial probability distributions of estimated event locations. To prepare the data for model training, we took the time sample where the probability for each individual operation onset is greatest as the ground truth, signifying the transition event location (see dotted lines in Figure 5).

The EEG data was epoched as part of preprocessing, meaning that each trial starts at stimulus onset. To prepare this data for use in machine learning models, we have to label it. For every trial, all samples between stimulus onset and the first observed event are labeled as *Operation 1*. The samples between the first and the second observed event are labeled as *Operation 2*, and so on until the last operation, which begins at the last observed event and ends at the response time. A visualization of this can be seen in Figure 5. The order and names of operations are taken from the literature on the task for which the preprocessing is done. This process results in added labels for each timestep containing the sequence of cognitive processing operations as a list of words. Epochs where the maximum probability sample of any onset $i$ occurs later than onset $i+1$ were discarded, ensuring temporal integrity of the operation sequence. An unordered sequence of events can occur since HMP only guarantees that the weighted means of the probability distribution are ordered, not the maximum probability. Finally, the data is split up into separate segments based on the labels, resulting in a dataset where each epoch is split up into an amount of segments that equals the amount of expected cognitive operations. Each segment represents a single cognitive processing operation and is zero-padded to the maximum single sample length in the dataset.

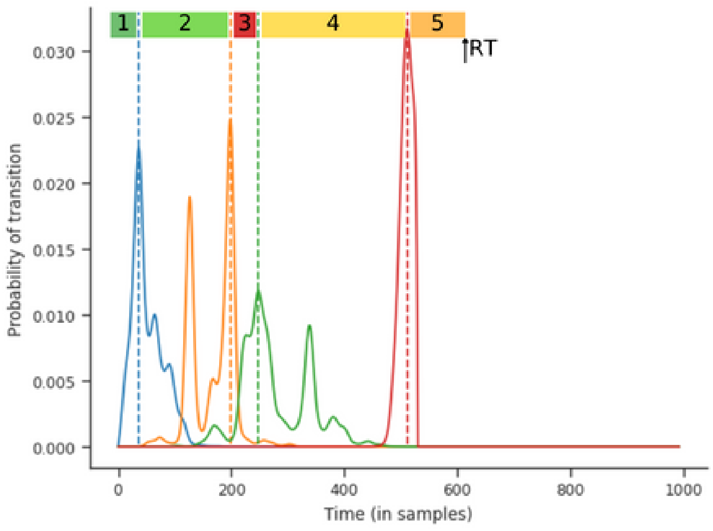

Figure 5: Probability distribution of transition event timings for a single trial, dashed vertical line denotes the maximum probability time sample. At the top a visual representation of processing operation segmentation is added, colors are chosen to be similar to Figure 3.

### 3.1.2 Training

The next step in the pipeline is made when the dataset is loaded for training. The dataset is split into partitions based on the participants, ensuring that for any participant that occurs in the train set, none of their data occurs in the test set. While splitting, the dataset is also optionally normalized (see Section 3.4.1). Here, normalization is done using information extracted only from the train set, to prevent information bleeding through from the train set to the test set. Splitting of the dataset is done in two different ways, the first is a simple train/test/validation split where 60% of participants belong to train, and 20% to both test and validation. This method of splitting was used for quick iteration and testing different model parameter configurations. K-fold cross validation was then applied to validate performance. For all parameter tests performed on one dataset, $k$ was set to the number of participants.

Since the datasets are saved in a sparse manner, any operation that occurs uniquely in one condition and not in another, is part of the dataset. For example, in the SAT1 and SAT2 datasets, the *confirmation* operation is also included as NaN values for trials under the 'speed' condition. We remove all of these trials along with any other trial for which the data is missing, due to for example electrode malfunction. This ensures that any remaining NaN values are padding values added by the last step of processing. We replace these by a masking value, to be ignored during training. Lastly, we shuffle the data to ensure that the model trains on a varying order of participants and operations. The data is served to the models in batches of 128 segments.

After processing the data and preparing it for training, it is now ready to feed into a model. The machine learning framework PyTorch (Paszke et al., 2019) was used to create the models. To train, PyTorch DataLoaders are employed to allow for fast parallel training. Run information is logged to event files supported by TensorBoard (Abadi et al., 2015). Class weights are calculated based on the amount of occurrences of each label and are given to the cross-entropy loss function. When applicable, the label smoothing parameter was used to regularize model predictions. The NAdam optimizer is used with weight decay (L2-regularization) when necessary. Training is stopped early (T. Zhang & Yu, 2005) if validation loss does not decrease for three subsequent epochs, at this point, the model is very unlikely to improve. After every training run, the epoch with the lowest validation loss is saved. To decrease variation between runs the random seeds were set before every run or fold. This was done for the python environment using `random.seed()`, NumPy using `np.random.seed()` and PyTorch using `torch.manual_seed()`.

Training was done on a desktop computer with a single NVIDIA RTX 3090 graphics card and 64 gigabytes of RAM.

## 3.2 Models

In this section, we introduce the machine learning models deployed to classify cognitive processing operations in EEG data. This section provides a comprehensive overview of the models used, highlighting their differences, functionalities, and the rationale behind their selection for this paper.

We begin with a detailed description of the CNN models used. This discussion focuses on the model's architecture and its ability to discern spatial patterns in EEG data. Following this, we examine RNN models and their more current variants, containing LSTM and GRU cells. Here, we discuss how these models' capabilities in processing sequential data make them particularly suitable for EEG data analysis, where temporal information plays a significant role. Lastly, we introduce the transformer model which is a more recent model architecture that encodes sequence information into internal embeddings and uses attention mechanisms which may prove important to achieve good generalization across contexts.

Each model's section will introduce their basic definition, some aspects or choices will be explained later based on parameter test outcomes. Model definitions are included in Appendix A. In this section, we will often use the term 'module', meaning a grouping of layers that can be stacked on top of each other in a model.

### 3.2.1 Convolutional neural network

Different variants of a CNN model were used to optimize for varying aspects of the data. The general reasoning for using a CNN model is to make use of the spatial information present in EEG data. An important challenge when using CNN models for incomplete data, like images with holes, or time series where part of the time series is uninformative, is making sure the model does not use this information (Liu et al., 2018). In this section, we will explain how the CNN models we employed handle this issue and explain the individual differences among the CNN models.

The base model works on input of shape $(batch\_size, samples, channels)$. All CNN models used require adding an extra dimension for the data itself, internally represented as $(batch\_size, 1, samples, channels)$. The base model consists of three modules, each containing a convolution layer, ReLU activation, and max pooling layer. In the first module, the convolution layer uses partial convolution (Liu et al., 2018) to ensure the model does not consider the masking values. The output of the final max pooling layer is then flattened and used in two linear layers, with ReLU activation and dropout in between. Convolutions in this model apply only to the temporal (samples) dimension, since the input data does not contain spatial information, meaning that convolutions would be done on spatially unrelated channels.

To aid in the data formulation parameter test, two other CNNs were created. Firstly the topological CNN, this model is the same as the base model, except that it works on input of shape $(batch\_size, samples, x, y)$. In this model the convolution still only works on the temporal dimension. Secondly, the topological convolution CNN was created, which also works on input of shape $(batch\_size, samples, x, y)$. This model includes convolutions over the $x$ and $y$ spatial dimensions.

In the parameter tests where data with a higher sampling frequency was used, deep variants of earlier models were employed. These have a similar setup, except that they have more modules and the kernel size of the convolutional layer is greater. These models make use of the higher temporal frequency and are meant to detect finer-grained patterns in the temporal dimension.

### 3.2.2 Recurrent neural network

Because RNNs (including LSTM, GRU models) were conceived to find longer-term relationships in data (Hochreiter & Schmidhuber, 1997), and EEG data is collected over a sequence of time, we included RNN models in our model comparison. EEG data is both spatially and temporally informative, but in this case the prediction is that an RNN may perform better than a CNN. Since cognitive operations are inherently sequential, models that perform well on sequential data could be a better fit.

Two RNN models were created to classify cognitive processing operations, we decided to focus on the two more modern variations of an RNN network, LSTM and GRU. RNN models differ from CNN models by the method they use to determine features. Where CNN models use convolution, filters, and pooling, an RNN model uses the information in a sequence of information, along with feedback connections, to learn relationships between different time points. LSTM and GRU models differ from an RNN in the mechanism they use to decide which information to keep.

The LSTM and GRU implementations are very similar, since they were used to compare the different RNN units available. They work on input of shape $(batch\_size, samples, channels)$, first masking out the padded values, applying the GRU/LSTM layer, followed by ReLU activation, and finally two linear layers. The prediction values at the last time sample for each segment are used to consider the collected internal representation.

### 3.2.3 Transformer

A transformer model was created inspired by the work of (Vaswani et al., 2017). The original transformer architecture was created for sequence-to-sequence modelling in natural language processing (NLP) tasks. A transformer model consists of an encoder which uses an attention mechanism to determine which parts of the input data are important. It embeds input data into embedding space, where a decoder is used to decode the information into another sequence of data. We expect a transformer model to perform well because of the aforementioned embedding space, which might improve generalizability across contexts, since a transformer compares within this space. We have adjusted the existing transformer architecture by removing the decoder part of the transformer model, creating a model that learns an embedding for incoming information and classifies based on this embedding. Since transformer models have many hyperparameters which can greatly impact their performance, we decided to stay as close as possible to proven configurations and leave hyperparameter tuning out of the scope of this paper since we only attempt to show a proof-of-concept.

The transformer model functions by first truncating segments in a batch, shaped $(batch\_size, samples, channels)$, to the maximum segment length in the batch. We then feed the remaining information through a fully connected layer, which is added to simulate a feature embedding layer in a traditional transformer and converts the sentences into embedding space. This addition should help the model learn more generalized feature representations. We then add a sinusoidal positional encoding as described in (Vaswani et al., 2017) to inform the model of the order included in the data. Afterwards a multi-head attention encoder layer receives the data, its output is mean-pooled over the temporal dimension and decoded into a final classification using a fully connected layer.

## 3.3 Performance

In this section, we discuss an experiment that tests the performance of the models and compares it to two simple baseline classifiers.

The performance was tested on two datasets, the 500 Hz SAT1 dataset and the 100 Hz SAT2 dataset. These were chosen since they provide two different views of the data. The 500 Hz SAT1 dataset offers more temporal information, whereas the 100 Hz SAT2 dataset provides more training samples.

We choose two relatively simple classifiers as our baseline, a random forest classifier (RF) (Breiman, 2001), and a support vector classifier (SVC) (Cortes & Vapnik, 1995). These were chosen for being simple to implement while also being able to learn non-linear relationships. Both RF and SVC use the default implementations offered by scikit-learn (Pedregosa et al., 2011).

To prepare the data for these classifiers, which do not by default handle 3-dimensional data, we calculate several numerical features. The mean, standard deviation, minimum, maximum, median and variance are calculated over the temporal dimension. These features are then aggregated over the channels dimension, creating data of shape $(batch\_size, \#channels \times \#features)$.

The models and their performance metrics were validated using $k$-fold cross-validation, where $k$ equals the number of participants in the dataset used.

## 3.4 Validation of (hyper)parameter choices

Here, the parameter validation tests that were used to determine which parameters work well for classifying EEG data within the context of this paper are described. We consider each of the parameter tests in turn, talking about the setup and hypothesis. These parameter tests were executed on a single dataset to find best practices for detecting cognitive operations from EEG using ML. Performance for each of the parameter tests was validated using $k$-fold cross-validation, where $k$ is equal to the number of participants of the experiment used.

### 3.4.1 Normalization

Machine learning models benefit from having all continuous features in the same numerical range (Santhakumaran, 2011), meaning that greater numeric values cannot overpower smaller ones. Even though epoching the data already makes it normalized between $[-55, 55]$, we expected the model to perform better at more commonly used ranges like $[0, 1]$, $[-1, 1]$ or Z-scored. Normalizing to a different range is done by subtracting the smallest value of the dataset from each data point, and dividing it by the smallest value subtracted from the largest value. Z-scoring is done by subtracting the mean and dividing by the standard deviation for each data point. To test which type of normalization performs best for EEG data, we compared the three discussed methods of normalization with not normalizing the data at all. The GRU and CNN models were used for this parameter test, trained on the 500 Hz SAT1 dataset.

### 3.4.2 Preprocessing

Manual preprocessing is a big part of EEG data analysis. In the case of the SAT1 dataset, individual component analysis (ICA) is used to divide the EEG data into components. These are then manually excluded based on if they include muscle spasms or eye blinks, or other artifacts (Weindel et al., 2021). If manual preprocessing steps were not necessary in EEG analysis, this would make a method more widely applicable since no expert is required for manual preprocessing. The goal of this parameter test is to determine if a model is still able to classify correctly on data that is not manually preprocessed. The GRU model was used for this dataset, along with a modified version of the SAT1 dataset, where preprocessing was replicated excluding the manual step of detecting and removing ICA components.

### 3.4.3 Sampling frequency

The temporal frequency at which EEG data is gathered differs across experiments. Many analysis methods downsample EEG data to 100 Hz for reasons of computational feasibility. Since the subject at hand is temporal in nature, we expect that using a higher sampling frequency dataset will result in better performance since more fine-grained temporal patterns can be extracted. To test this we compared performance of both CNN and GRU models trained on a 100 Hz dataset and a 500 Hz dataset. The SAT1 dataset was used. The source dataset is 500 Hz, to create the 100 Hz dataset we first downsampled the EEG data to 100 Hz, then trained an HMP model on the resulting data. Finally, we compared performance across the trained models to determine the effect of sampling frequency on model performance.

### 3.4.4 Formulation

The way that data is shaped can have an effect on the performance of a model (Altaheri et al., 2021), in the case of EEG data the distribution of activity over the scalp may be informative. A CNN learns from spatial relationships, meaning that the input data should be spatially informative. In this parameter test, we re-shape the 2-dimensional $(samples, channels)$ data into a 3-dimensional $(samples, x, y)$ formulation. The positions of the channels within the $x$ and $y$ dimension correlate to the electrode positions on the scalp (see Figure 6).

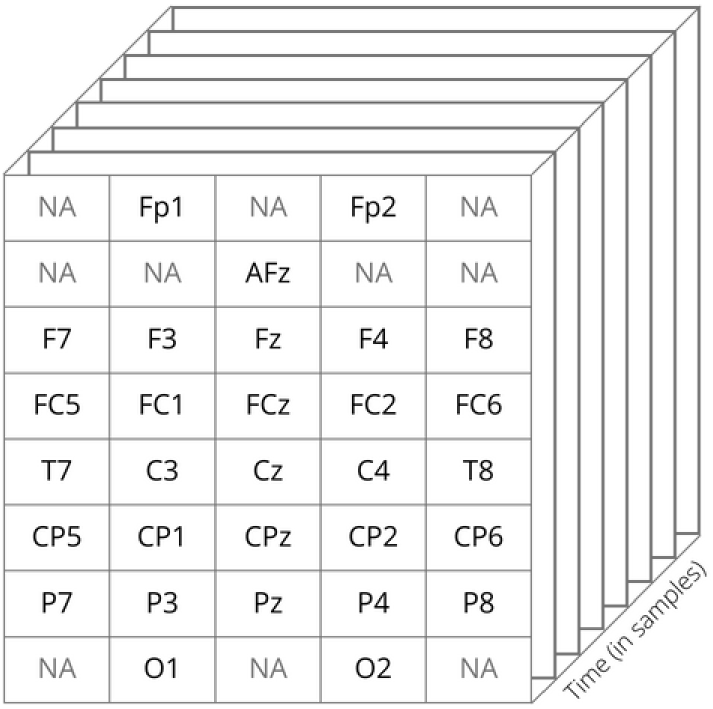

Figure 6: Layout of electrodes in a sparse 3-D formulation.

We expect that the CNN model used in this parameter test will perform better when the data is formulated in 3 dimensions. We compare the base CNN model with a CNN model using 3-dimensional formulation without convolution over the added dimensions and the same model with convolution over the added dimensions. The 100 Hz SAT1 dataset was used.

## 3.5 Generalization

We will now discuss the methodology used in exploring the generalizability of the different models across different contexts. We define generalizability in this case as the difference in performance of a model that is trained on dataset $x$ and tested on dataset $x$, versus that same model tested on dataset $y$. The context of datasets $x$ and $y$ is what differs, like condition, lab, or experimental task setup.

The initial parameter tests discussed above revealed that the SAT1 dataset was too small for the transformer model to learn useful features. Therefore, we decided to train the models on the SAT2 dataset, which is much larger. Firstly we train a model on the SAT2 dataset's accuracy condition, then we test that model on the speed condition to show generalization across conditions. Secondly, we train models on the full SAT2 dataset and test the models on the SAT1 dataset and the AR dataset. This shows generalization across both labs and tasks. Training and testing were done on 100 Hz versions of the datasets, as the AR dataset is not available at a higher temporal resolution. When processing the additional datasets, we truncate the samples in those datasets to the maximum segment length in the original dataset. Additionally, because the transformer model did not learn well without some regularization techniques added. We set weight decay (L2-regularization) to 0.001 in the optimizer, and label smoothing to 0.0001 in the loss function. We did this for all models.

## 3.6 Feature visualization

To visualize what the models learn, and what, according to the model, makes a processing operation unique, we applied feature visualization to the models. We used the integrated gradients (Sundararajan et al., 2017) method applied to the test set of a trained model. This gives us the importance of each channel at each time point for a certain prediction. To more closely inspect these feature attributions across both dimensions, we introduce two methods to visualize them. We use the implementation of integrated gradients provided by (Kokhlikyan et al., 2020). We set the baselines to 0 for every non-masked value as this can also be seen as the 'no-activity' value in EEG data. We calculate 50 gradient steps and use the Riemann trapezoid method as implemented in (Kokhlikyan et al., 2020).

For the spatial dimension, we take the subset of attributions belonging to a single class and calculate the activation per time point summed over channels for each segment. The time point where the summed activation is greatest is selected as the time point that is most indicative of the prediction. The average of all of these maximally activated time points is visualized as the raw EEG data and the model attribution at that time. Additionally, these are combined by multiplying the raw EEG data with the model attribution, creating a visualization showing what levels of EEG data are important at which location for the model to classify a segment as a given class.

In the temporal dimension, we face the problem of variable length processing operations. To circumvent this we linearly interpolate the model attributions for each class and average over these interpolations to show at what point in a stage the model attention is greatest. To make the interpolation more robust, we filter out segments with a length shorter than 3 samples.

## 4 Results

This section presents the findings from our investigation into classifying cognitive processing operations using machine learning techniques. We analyzed the obtained EEG datasets to determine whether a trained classifier is able to recognize similar processing activity across different datasets. We will begin with discussing the performance of the introduced models compared to baseline classifiers. Afterwards, we examine the findings from the parameter validation tests and what their results imply for applying machine learning to EEG data. Finally, we present the generalization results obtained by comparing model performance across datasets.

### 4.1 Performance

The results of the performance tests show that the model architectures introduced in this paper outperform the selected baseline classifiers. To show differences between the distributions of performance metrics over folds, we use violin plots. A difference matrix is provided to more closely inspect the differences in performance between each classifier type. The color blue represents accuracy in both the violin plot and the difference matrix, orange represents F1-score. The difference matrices are read in two ways, accuracy is read from the x-axis to the y-axis, and the F1-score from y-axis to x-axis. This method ensures that the intensity of the cell conveys the same meaning across both metrics. Tables of mean metric values along with their standard deviation are also provided for exact values.

The results for the 100 Hz SAT2 dataset as shown in Figure 7 show that the GRU model performs best on this dataset. When compared to the performance results for the 500 Hz SAT1 dataset (visualization of results included in Appendix B.2), we can observe that the SVC, CNN, and transformer models benefit more from added training samples than a higher temporal resolution. The GRU model delivers a similar performance on the 500 Hz dataset as on the 100 Hz dataset with more training samples. This indicates that a higher temporal frequency provides a performance in improvement for the GRU model. Additionally, we can see that the GRU models have a more peaked distribution and a lower standard deviation than any other model, meaning that the variability of performance over folds is lowest.

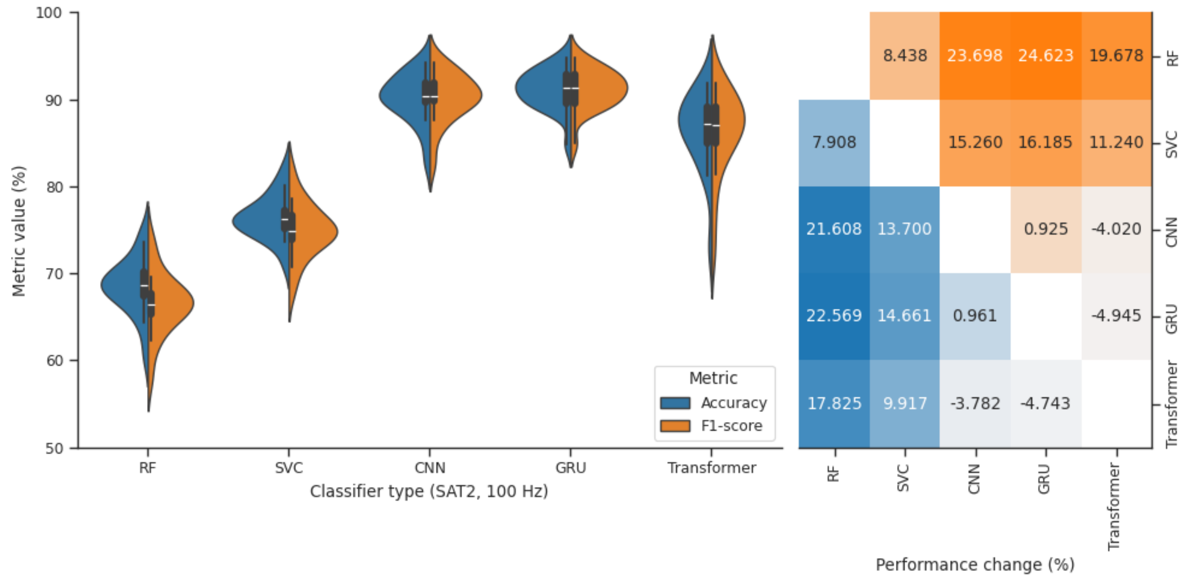

Figure 7: Results of performance test on SAT2, 100 Hz dataset. The GRU model performs best overall on this dataset. Left panel: accuracy/F1-scores for each classifier type. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference between classifier types.

| Classifier type | Dataset | Accuracy | F1-Score |
|---|---|---|---|
| RF | SAT2, 100 Hz | 68.57% (SD 3.14) | 66.51% (SD 3.41) |
| SVC | – | 76.48% (SD 2.49) | 74.95% (SD 3.04) |
| CNN | – | 90.18% (SD 2.74) | 90.21% (SD 2.77) |
| GRU | – | **91.14%** (SD 2.29) | **91.13%** (SD 2.29) |
| Transformer | – | 86.39% (SD 4.43) | 86.19% (SD 4.78) |
| RF | SAT1, 500 Hz | 66.03% (SD 7.34) | 65.02% (SD 8.06) |
| SVC | – | 69.72% (SD 5.91) | 69.03% (SD 6.52) |
| CNN | – | 88.11% (SD 5.82) | 88.12% (SD 5.77) |
| GRU | – | **91.73%** (SD 3.55) | **91.72%** (SD 3.54) |
| Transformer | – | 76.44% (SD 7.66) | 76.26% (SD 7.82) |

Table 1: Performance of baseline classifiers compared to models from the paper on the SAT2 100 Hz and SAT1 500 Hz datasets. Bolded numbers indicate the highest performance metrics. Dashes represent repetition of the above column.

## 4.2 Validation of (hyper)parameter choices

Here, we report the most important results from each parameter validation test. To show differences between the distributions of performance metrics over folds, we use violin plots. A difference matrix is provided to more closely inspect the differences in performance between each parameter choice.

### 4.2.1 Normalization

In this parameter test, we attempted to understand how different normalization methods affect the performance of a classification model on EEG data. We compare the distributions of accuracy and F1-score metrics across four normalization techniques 'Control', '0 to 1', '-1 to 1', and 'Z-score'.

The results show that the differences are not very large across normalization methods. The violin plots reveal that the '-1 to 1' normalization method, normalizing to the range $[-1, 1]$, generally leads to higher metric values compared to the control method. The more pronounced peak in the distribution suggests a tighter distribution of scores. In contrast, the 'Z-score' method shows a broader distribution, indicating more variability in performance.

The same parameter test for the CNN model is included in Appendix B.1, where the results are similar but show that the '0 to 1' method underperforms for the CNN model. These results show that the normalization method is not something that can be set in stone. When performance is important, the normalization method should be tested in combination with each dataset and each model architecture, to find the best performing normalization method under the circumstances.

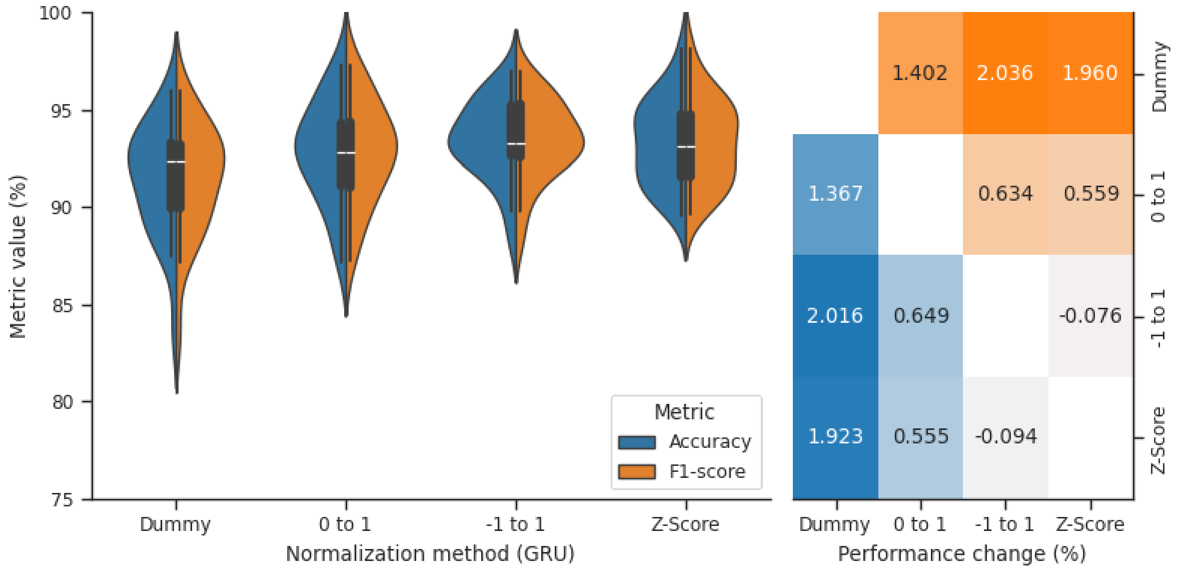

Figure 8: Results of normalization parameter test. Normalizing to the range $[-1, 1]$ is optimal. Left panel: accuracy/F1-scores for each normalization method applied. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference between normalization methods.

| Normalization method | Accuracy | F1-Score |
|---|---|---|
| Dummy | 91.46% (SD 2.78) | 91.42% (SD 2.82) |
| 0 to 1 | 92.82% (SD 2.58) | 92.82% (SD 2.56) |
| -1 to 1 | **93.47%** (SD 2.01) | **93.46%** (SD 2.01) |
| Z-Score | 93.38% (SD 2.17) | 93.38% (SD 2.16) |

Table 2: Performance metrics of normalization parameter test (GRU). Bolded numbers indicate the highest performance metrics.

#### 4.2.2 Preprocessing

This parameter test was designed to evaluate the influence of manual data preprocessing on the accuracy and F1-score of a classification model. We compare an unprocessed dataset against the control, our processed dataset.

Although the performance metrics do not differ by much, the plots reveal that the 'Control' condition shows a slightly more compact distribution of performance metrics, as indicated by the narrower shape of its plot. This implies that manual preprocessing generally improves the certainty of the model.

Interestingly, the difference matrix shows that performance across both metrics increased when moving to the unprocessed dataset. We think this is because the step of removing ICA components related to eye-blinking or muscle spasms includes a human decision, introducing uncertainty into the dataset in both the decision-making and the ICA algorithm itself.

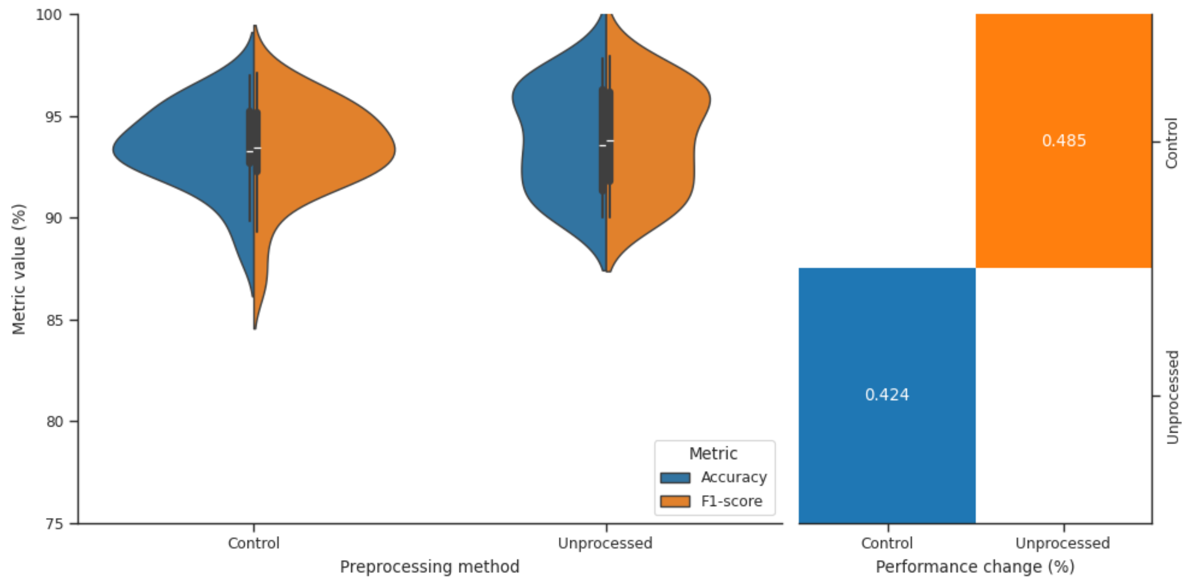

Figure 9: Results of preprocessing parameter test. Using unprocessed data does not decrease performance. Left panel: accuracy/F1-scores for the preprocessing methods. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference between preprocessing methods.

| Preprocessing method | Accuracy | F1-Score |
|---|---|---|
| Control | 93.47% (SD 2.01) | 93.39% (SD 2.20) |
| Unprocessed | **93.90%** (SD 2.44) | **93.88%** (SD 2.46) |

Table 3: Performance metrics of preprocessing parameter test. Bolded numbers indicate the highest performance metrics.

### 4.2.3 Sampling frequency

In this parameter test we show the difference in performance across model types and sampling frequency. We compare the sampling frequencies 100 Hz and 500 Hz for two types of models, CNN and RNN (GRU).

What the results show is that at a higher frequency, performance increases for all metrics and model types. The distributions are also more peaked at a higher frequency, implying that the model is generally more certain.

The combination of the violin plot and the performance matrix also shows that the GRU model improves more by increasing the sampling frequency than the CNN model. This implies that GRU models are more sensitive to changes in sampling frequency than CNN models.

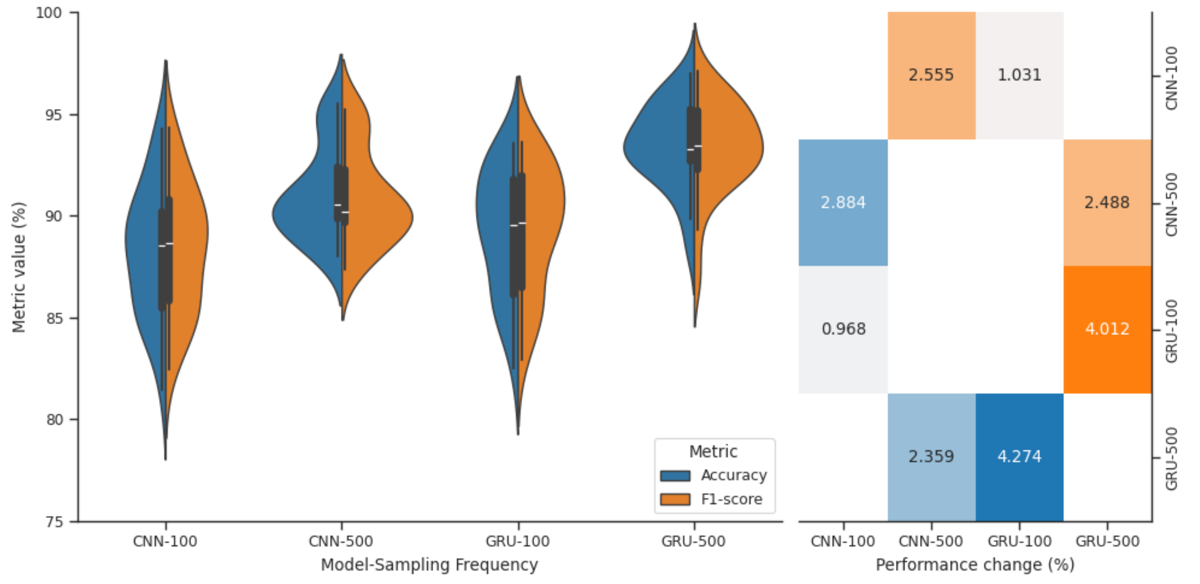

Figure 10: Results of sampling frequency parameter test. Some differences left out since comparing across model type and sampling frequency is not informative. GRU models improve more by increasing sampling frequency. Left panel: accuracy/F1-scores for each sampling frequency and model combination. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference between sampling frequency and model combinations.

| Model | Sampling frequency | Accuracy | F1-Score |
|---|---|---|---|
| CNN | 100 Hz | 88.23% (SD 3.18) | 88.35% (SD 3.11) |
| CNN | 500 Hz | 91.12% (SD 2.25) | 90.91% (SD 2.29) |
| GRU | 100 Hz | 89.20% (SD 3.04) | 89.38% (SD 3.03) |
| GRU | 500 Hz | **93.47%** (SD 2.01) | **93.39%** (SD 2.20) |

Table 4: Performance metrics of sampling frequency parameter test. Bolded numbers indicate the highest performance metrics.

### 4.2.4 Formulation

In this parameter test, we compare three different methods of data formulation: Control, which is $(batch\_size, samples, channels)$. Topological, which is $(batch\_size, samples, x, y)$ without convolution over the new $x$ and $y$ dimensions, and topological+convolution, which is $(batch\_size, samples, x, y)$ with convolution over the new $x$ and $y$ dimensions added.

The results suggest that introducing the added complexity of a new dimension does not outweigh the supposed gain from laying the data out in a spatially informative formulation. This can mean two things: a CNN model finds spatial relations even without a spatially informative formulation, or the multivariate nature of patterns found in EEG data is difficult to detect by using regular convolution. We will return to this point in the discussion.

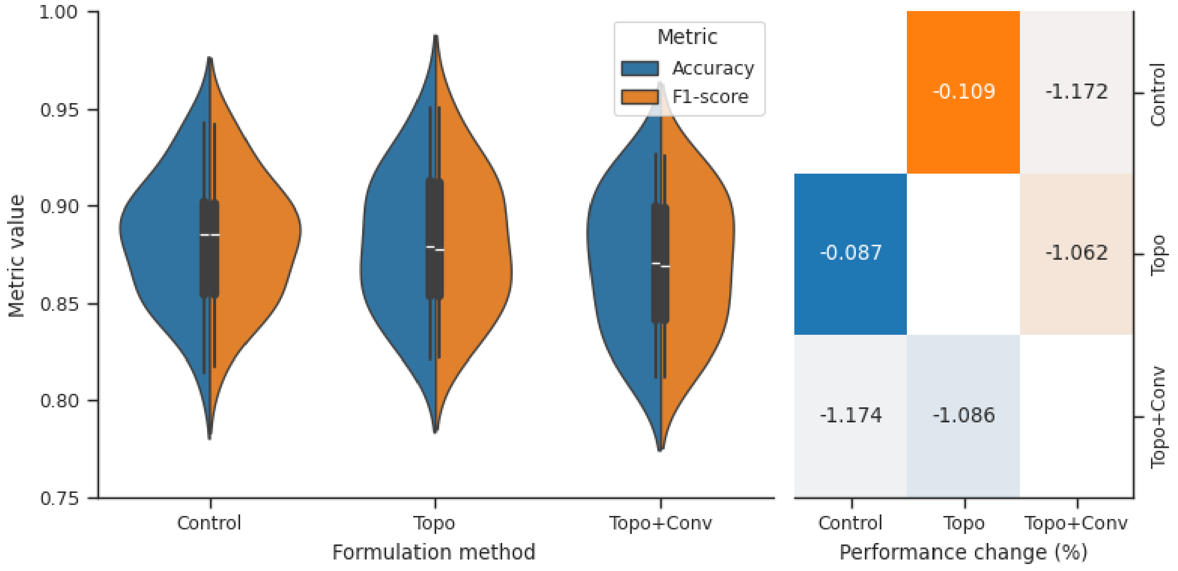

Figure 11: Results of data formulation parameter test. Adding spatial information to the CNN model does not increase performance. Left panel: accuracy/F1-scores for each formulation method. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference between formulation methods.

| Formulation method | Accuracy | F1-Score |
|---|---|---|
| Control | **88.23%** (SD 3.18) | **88.20%** (SD 3.15) |
| Topo | 88.14% (SD 3.49) | 88.09% (SD 3.46) |
| Topo+Conv | 87.06% (SD 3.50) | 87.03% (SD 3.42) |

Table 5: Performance metrics of data formulation parameter test. Bolded numbers indicate the highest performance metrics.

## 4.3 Generalization

In this section, we show to what extent the trained models generalize to other contexts. We again use violin plots to show differences in distributions across conditions. Difference matrices are included to more closely inspect the differences in values.

### 4.3.1 Across conditions

What these results show is that each model class performs similar when generalizing across conditions, the violin plots in Figure 12, Figure 18, and Figure 19 (last two included in Appendix B.3) show that there seems to be a few participants for which the accuracy condition does not inform the speed condition. This explains the higher standard deviation for the speed condition. Upon investigation, we observe two participants where this occurs. In the case of the GRU model, one performs at or around chance (accuracy: 23.29%, F1-score: 20.52%), the other has an accuracy of 37.40% and an F1-score of 35.28%. We can clearly see that the CNN model loses more performance than the GRU and transformer models. Since the GRU model starts and ends at a higher performance we can say that the GRU model generalizes best across condition, within the constraints of the given configuration and dataset.

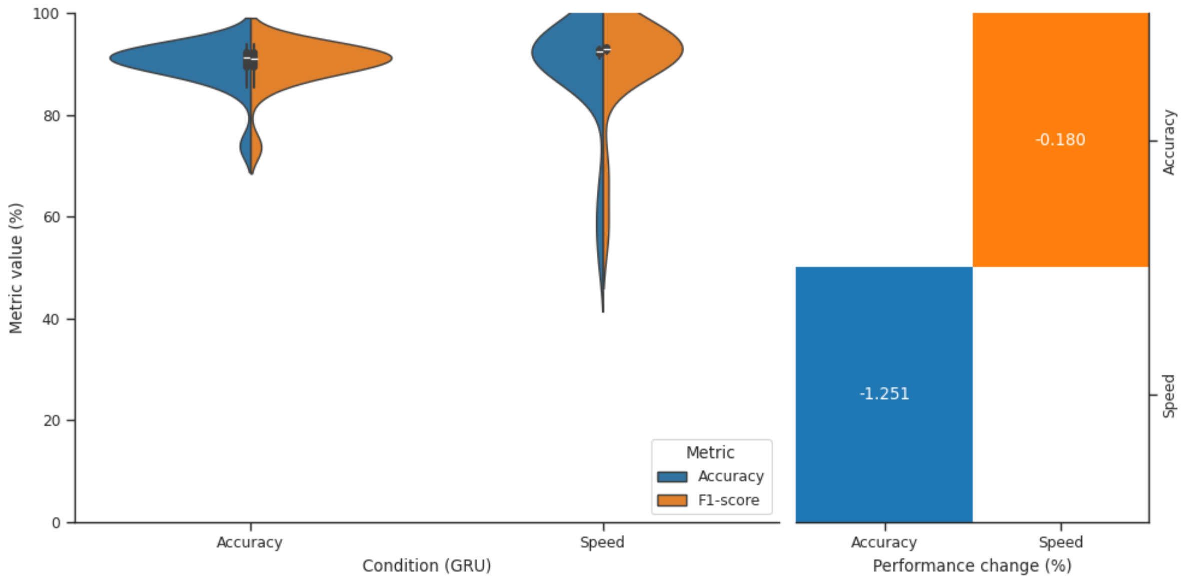

Figure 12: Generalization across conditions in a GRU model. Model trained and validated on accuracy condition, tested on speed. The GRU model is able to generalize across conditions. Left panel: accuracy/F1-scores for each condition. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference across conditions.

| Condition | Dataset | Accuracy | F1-Score |
|---|---|---|---|
| Accuracy | GRU | **89.94%** (SD 4.46) | **89.92%** (SD 4.49) |
| Speed | – | **88.69%** (SD 10.77) | **89.74%** (SD 9.67) |
| Accuracy | CNN | 89.85% (SD 2.82) | 89.83% (SD 2.83) |
| Speed | – | 85.53% (SD 19.66) | 85.92% (SD 20.67) |
| Accuracy | Transformer | 86.22% (SD 5.22) | 85.77% (SD 6.57) |
| Speed | – | 84.63% (SD 20.07) | 86.04% (SD 20.33) |

Table 6: Performance metrics of models generalized across conditions. Bolded numbers indicate the highest performance metrics. Dashes represent repetition of the above column.

### 4.3.2 Across labs & task

The results visualized in Figure 13 (results for CNN and Transformer model included in Appendix B.4) demonstrate that the models as implemented in this paper are somewhat able to generalize across labs, showing a decrease in performance of around 30% for generalizing from the SAT2 dataset to the SAT1 dataset. The results also show that the models are not able to generalize across tasks, as the performance of the classifiers tested on the AR dataset is not above chance level (20%).

All models fail to generalize to the SAT1 dataset for one participant, performing around chance level (accuracy: 29.19%, F1-score: 25.01%, performance metrics taken from the GRU model), while the models performed well on that participant in the SAT2 dataset (accuracy: 93.87%, F1-score: 93.85%). This participant is the same as one of the two participants for which the same occurs when generalizing across conditions.

The transformer model starts off with a lower average performance, with higher variance, and also loses more performance when generalizing. This indicates that the transformer model as implemented is the worst at generalization. The GRU model performs equal or slightly worse than the CNN model on generalization across labs, but retains a bit more performance when generalizing across tasks.

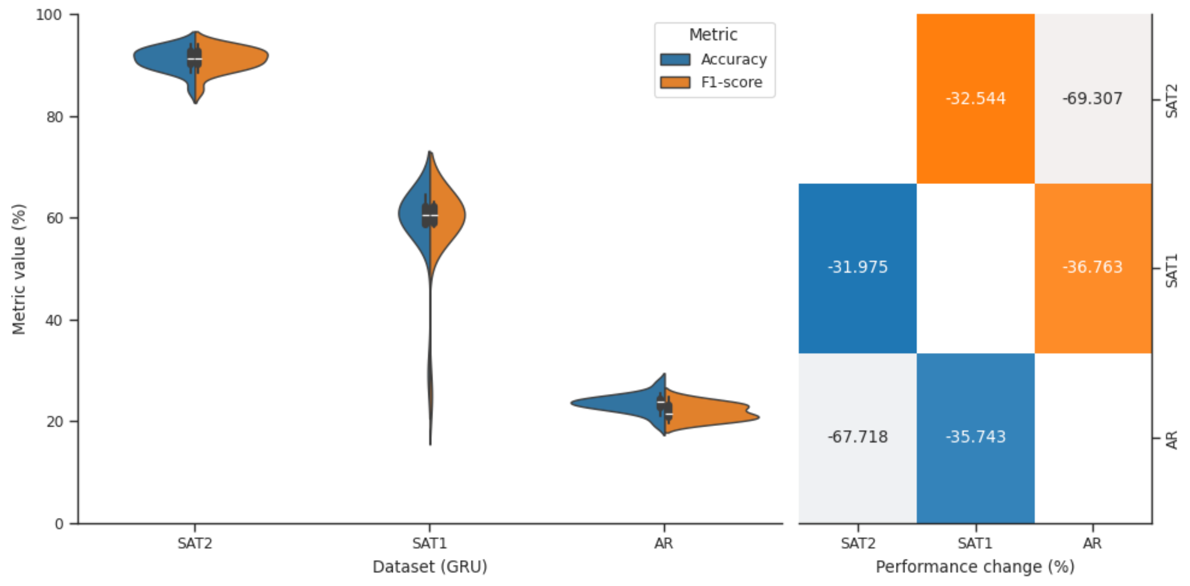

Figure 13: Generalization across labs & tasks in a GRU model. The GRU model is somewhat able to generalize across labs and not across tasks. Left panel: accuracy/F1-scores for each dataset. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference across datasets.

| Dataset | Accuracy | F1-Score | |
|---|---|---|---|
| SAT2 | GRU | **91.17%** (SD 2.21) | **91.18%** (SD 2.22) |
| SAT1 | – | 59.20% (SD 7.53) | 58.63% (SD 8.32) |
| AR | – | **23.46%** (SD 1.73) | 21.87% (SD 1.55) |
| SAT2 | CNN | 90.50% (SD 2.49) | 90.51% (SD 2.50) |
| SAT1 | – | **61.12%** (SD 8.34) | **60.12%** (SD 8.97) |
| AR | – | 16.48% (SD 4.32) | 13.59% (SD 3.78) |
| SAT2 | Transformer | 86.26% (SD 3.56) | 86.16% (SD 3.73) |
| SAT1 | – | 49.13% (SD 8.66) | 46.25% (SD 10.67) |
| AR | – | 20.93% (SD 4.10) | **22.03%** (SD 2.72) |

Table 7: Performance metrics of models generalized across lab (SAT1) and task (AR). Bolded numbers indicate the highest performance metrics. Dashes represent repetition of the above column.

### 4.4 Feature visualization

We have so far shown that models are able to generalize in some fashion to unseen data. But what is it that the models are learning? To show this we applied integrated gradients. Shown in Figure 14 are the visualized features for each of the different processing operations. These graphs are included for the CNN and GRU models in Appendix C. The values calculated by using integrated gradients in the second row are indicative in both directions, so negative (blue) values are also meaningful. Investigation of the visualizations shows that what the model finds important to distinguish the processing operations differs among the models. The transformer model as shown here attends to similar areas for the *pre-attentive*, *confirmation*, and *response* stages, but the activity it attends to is changing, as seen in the differences in the *combined* row. Activity in the occipital area is most important for the model to distinguish the *pre-attentive*, *encoding*, and *decision* stages, while activity in the parietal area is most important for the *confirmation* and *decision* stages.

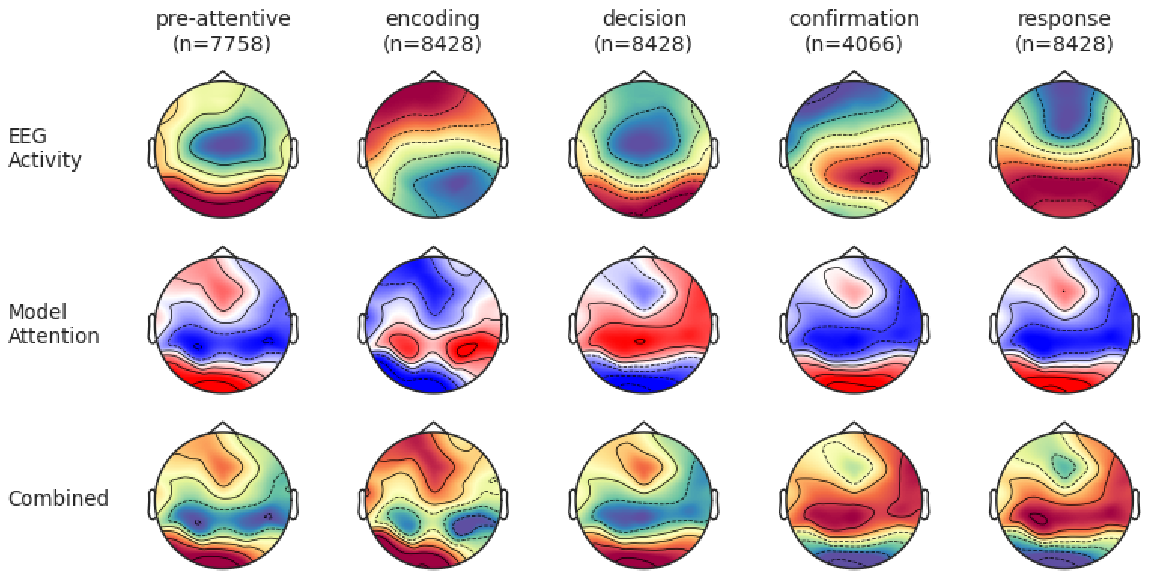

Figure 14: Integrated gradients visualization for the transformer model.

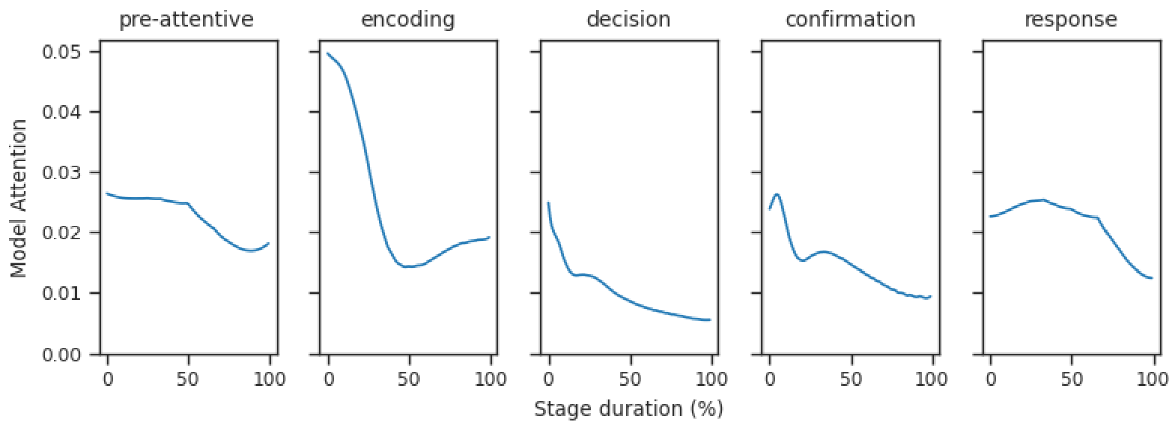

Figure 15: Interpolated model attention over stage duration for the transformer model. Y-axis zoomed in compared to CNN and transformer, original found in Appendix C.3

## 5 Discussion

The goal of this paper was to show that similar cognitive operations occur in different contexts, as defined by their EEG representation. Moreover, we aimed to train ML classifiers to identify these representations in unseen data, as a way to substantiate the similarity across contexts.

The results show that ML models are able to recognize and tell apart different cognitive processing operations, but that they are better at recognizing operations in contexts closer to the context they were trained on. These results serve a couple purposes:

Firstly, they validate HMP as a method used to analyze EEG data. The models are consistently able to discern the segments of EEG data between HMP-suggested per-trial event timings, meaning that there is something that differs between them. There exists a possibility that the models would perform equally well during training using a random segmentation. However, random segmentation should perform at chance level across a sufficiently large test set, as no information about the structure of the data is incorporated in the segmentation. Because our models perform above chance, we think it is unlikely that random segmentation would perform similarly. This intuition could be tested by comparing performance on the HMP-provided segmentation with random segmentation, this was left out of the scope of this paper.

Secondly, the presented results serve as a starting-point for further research, showing that generalization is possible in principle. Further research could for example investigate a dual-task experiment paradigm. The findings from this paper would be used to detect cognitive processing operation in two single tasks, and verifying whether these operations can be found in data from the dual-task execution. Often, these dual-task experiments have been executed with the assumption that participants go through the same operations (Van Maanen & Van Rijn, 2010; Van Maanen et al., 2009), but this has not been validated.

Another interesting research direction is to investigate the types of processing operations that occur in cognitively impaired individuals. Previous research has shown that reaction time increases when tasks incorporate areas of the brain that are impacted by cognitive impairment (Anders et al., 2017; Winkel et al., 2016). We could use HMP to split EEG data collected from these individuals into segments, and then predict which cognitive operation those segments represent. If we find that a certain stage differs from a control group, or is missing, we could focus on clinical interventions on brain functioning connected to that processing operation.

Finally, we can develop a method which can be used to validate theories on processing operation sequence and duration in unseen data. Often, assumptions are made about these, but with this method, we can examine each processing operation for its similarity to the operations known to the model. This can open new avenues for research if we for example find that the sequence of operations is different for a certain subset of participants, what makes this group different?

A major assumption that was made in this paper is that a processing operation has a defined end. That is, we assume strict seriality in processing (Anderson, 2007; Atkinson & Shiffrin, 1968;

Sternberg, 1998). This assumption is not supported by research on parallel processing (McClelland, 1979) and has more recently been suggested to be incorrect (Dubarry et al., 2017; Oberauer & Kliegl, 2004; Stanton, 2002). As a consequence of the strict seriality assumption, we decided to end the segment at the time sample before the next onset. This assumption was made based on the idea that besides the event timing and topology, there could be more information hidden away in the time between two onsets that would assist in correctly classifying a cognitive operation. Investigating if this assumption is true was outside the scope of this paper but would be an interesting research direction. Varying the segment length and investigating classifier performance could provide valuable insights. At which size does the classifier find the greatest variance between classes? What is the difference in features learned? A downside of this approach is that we are then defining a cognitive processing operation as only the activity at its beginning. Our intuition was that a sufficiently complex model might be able to discern the activity of one processing operation from another, while they are executed in parallel.

Testing generalization showed that performance on other datasets had much greater variability over folds. This indicates that the models are not consistently able to learn general features which transfer to other datasets. We interpret this not as a failure of the model, but more as a potential consequence of the EEG data. Differences in experimental contexts such as the configuration of the Faraday cage or the exact placement of electrodes can introduce noise that seemingly makes it difficult for ML models to match its learned features to this data. We believe that introducing techniques used in the research areas of out-of-distribution generalization (Gagnon-Audet et al., 2023) and domain generalization (Pohjonen et al., 2022) could improve generalization performance. Perhaps a first step should be to examine the data we have from different perspectives, comparing for example the different frequency bands in the processing operation segments, to find if similarities can be found without using a ML classifier. Methods of dimensionality reduction like t-SNE (Maaten & Hinton, 2008) or UMAP (McInnes et al., 2020) could also be applied to the data to cluster similar processing operations and provide a more informed foundation for how to tune a ML model to cognitive processing operations.

In the generalization tests, some specific participants failed to generalize, performing around chance level. When generalizing across condition, one participant's data showed this behavior. When generalizing across lab, two participants showed this behavior, of which one is the same as the previously mentioned participant. Had all these participants been different, we could have argued that these results may just be a consequence of the combination of randomly chosen seed, optimizer, and loss function in combination with the exact subset of data. However, since this occurs across both types of generalization, we believe that there may be something to do with either the EEG recordings for this participant, or their behavior. Perhaps their strategy for the speed condition is substantially different than other participants, making the HMP timings for this data unreliable. The easiest way to solve this issue is to disregard this participant entirely, presumably resulting in better HMP event timings and thus a better performing model, since variance in the data has decreased. We think that examining what is different about this participant behaviorally offers a better approach, since it could teach us more about individual differences in task execution.

This paper did not consider data augmentation, which is a proven method in improving generalization (Lashgari et al., 2020). We believe that methods that create new samples based on existing data such as noise addition or using a generative model that generates additional training samples could make the model more robust to test data from different contexts. Since the samples generated using a generative model trained on only that dataset will still resemble data from the original dataset, we think that methods that augment existing data to resemble out-of-distribution data are more promising.

The parameter tests undertaken during this paper were aimed at identifying model parameters that performed best on the specific datasets used and should not be taken as broadly applicable best practices. Further research is required that tests the hypotheses used in the parameter tests across multiple datasets. That said, the results of the preprocessing parameter test show that manual preprocessing of EEG data is not necessarily required, which creates the possibility of online cognitive processing operation prediction. However, HMP is currently needed to segment the EEG data. The methods introduced do not require a HMP-created segmentation, but can function on any method that segments EEG data. The sampling frequency parameter test confirmed our intuition that an RNN would learn more from additional temporal information than a CNN.

We were surprised by the results of the data formulation parameter test, where we expected the added spatial information to increase performance. We believe that a CNN might not be able to learn features in EEG data using convolution, as the relationships between electrodes in EEG data differ from the relationships between pixels in image data. Convolution works well for finding contrast, edges, and more abstract features, but perhaps not for finding more complex multivariate relationships between EEG electrodes. The base CNN model used in this paper included convolution along the temporal dimension, which enabled it to find frequency patterns and amplitude variations that likely included enough distinctive information to create a classifier that performed well. We think that the temporal information was more indicative of the prediction than the spatial information added by introducing convolution over the new dimensions. Perhaps this would change when working with different neuroimaging technologies with a higher spatial resolution, like fMRI.

Feature visualization was done for the temporal and spatial dimensions separately, but this does not paint the whole picture, as the activity at one point in time can be important for a model to make a classification only in combination with activity at another point in time. We think a visualization method that combines both dimensions could provide valuable insights into what makes each processing operation unique. The simplest version of this would be a sequence of heatmaps, at certain intervals throughout the cognitive processing operation. This would solve the problem of examining only the single most important sample for a prediction, except that it does not provide any information on how the activation patterns across time are related. To solve this, we would first need to use a feature attribution method that considers the temporal dimension. The works of (Enguehard, 2023) and (Höllig et al., 2023) on time series interpretability methods could be used to create more interpretable feature attributions.

In this paper, the models that were used were left relatively simple, as the comparison between them was deemed most important. Each of these models can be further adjusted by for example grid search, or by adding techniques aimed at improving generalization. This paper does not provide a conclusive answer on which model is 'best' at generalizing for the purpose of classifying cognitive processing operations. It does show that each model is able to generalize, meaning that model choice should be made based on the dataset used. However, we believe that there is the most room to grow for the transformer model, as it showed considerable improvement when giving it more training samples. It is also the model that was most challenging to create a working version of, so along with having many configurable hyperparameters, it may hold the most promise as a model that is able to learn a diverse set of cognitive processing operations. Another method of increasing performance given the models we have, is to build a combined CNN and RNN model, which first learns a representation of the data at each time step using a CNN, and then feeds that into an RNN.

## 6 Conclusion

Our paper has shown progress in the fields of cognitive neuroscience and machine learning. While applying machine learning to EEG data has been done before, we have addressed a relatively untapped area of cognitive neuroscience, the classification of cognitive operations using machine learning classifiers on EEG data. Our paper addresses the central question: To what extent can machine learning models accurately classify cognitive processing operations, and how generalizable is such a classifier across different contexts or populations?

We have shown that an ML model is able to accurately classify cognitive processing operations. Besides that, they have the ability to decode operations in unseen data without loss of performance within condition, and with above-chance performance across lab. In the future, with models fine-tuned for generalizing to EEG data from different contexts, we believe that the same generalization performance could be reached for each of the contexts used in this paper. A model that is able to decode cognitive processing operations at such a level could be of use to expand our knowledge of cognition, assist experiment designers in validating task execution for participants, and potentially lead to a better understanding of cognitive disorders.

Our approach has initiated new avenues for understanding how the brain handles tasks through a series of cognitive operations. This has allowed us to acquire more certainty about the order and kind of cognitive operations taking place in different tasks. The use of EEG as the primary source of information underscores the importance of neuroimaging techniques in revealing patterns in brain activation related to cognitive processing operations.

This paper has leveraged various machine learning algorithms, including CNNs, RNNs, and transformers, to analyze spatial and temporal patterns in EEG data, enhancing our understanding of cognitive operations. The developed methodological framework, supported by a flexible data gathering, processing, and model training pipeline, has been instrumental in quickly adapting to new insights and exploring new questions in cognitive neuroscience.

## 7 References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., . . . Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/

Afshine Amidi & Shervine Amidi. (n.d.). *CS 230 - Convolutional Neural Networks Cheatsheet*. Retrieved June 19, 2023, from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

Altaheri, H., Muhammad, G., Alsulaiman, M., Amin, S. U., Altuwaijri, G. A., Abdul, W., Bencherif, M. A., & Faisal, M. (2021). Deep learning techniques for classification of electroencephalogram (EEG) motor imagery (MI) signals: A review. *Neural Computing and Applications*, *35*(20), 14681–14722. https://doi.org/10.1007/s00521-021-06352-5

Anders, R., Riès, S., Maanen, L. V., & Alario, F.-X. (2017). Lesions to the left lateral prefrontal cortex impair decision threshold adjustment for lexical selection. *Cognitive Neuropsychology*. Retrieved January 30, 2024, from https://www.tandfonline.com/doi/full/10.1080/02643294.2017.1282447

Anderson, J. R. (2007, October 1). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press. https://doi.org/10.1093/acprof:oso/9780195324259.001.0001

Anderson, J. R., Fincham, J. M., Qin, Y., & Stocco, A. (2008). A central circuit of the mind. *Trends in Cognitive Sciences*, *12*(4), 136–143. https://doi.org/10.1016/j.tics.2008.01.006

Anderson, J. R., Zhang, Q., Borst, J. P., & Walsh, M. M. (2016). The discovery of processing stages: Extension of Sternberg's method. *Psychological Review*, *123*(5), 481–509. https://doi.org/10.1037/rev0000030

Atkinson, R. C., & Shiffrin, R. M. (1968, January 1). Human Memory: A Proposed System and its Control Processes. In K. W. Spence & J. T. Spence (Eds.), *Psychology of Learning and Motivation* (pp. 89–195, Vol. 2). Academic Press. https://doi.org/10.1016/S0079-7421(08)60422-3

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016, July 21). *Layer Normalization* (1). arXiv: 1607.06450 [cs, stat]. Retrieved January 10, 2024, from http://arxiv.org/abs/1607.06450

Baddeley, A. D., & Hitch, G. (1974, January 1). Working Memory. In G. H. Bower (Ed.), *Psychology of Learning and Motivation* (pp. 47–89, Vol. 8). Academic Press. https://doi.org/10.1016/S0079-7421(08)60452-1

Bahdanau, D., Cho, K., & Bengio, Y. (2014, September 1). *Neural Machine Translation by Jointly Learning to Align and Translate*. arXiv.org. Retrieved June 19, 2023, from https://arxiv.org/abs/1409.0473v7

Berberyan, H. S., Maanen, L. van, Rijn, H. van, & Borst, J. P. (2020, August 28). *EEG-based identification of evidence accumulation stages in decision making*. https://doi.org/10.31234/osf.io/nmg6w

Berberyan, H. S., Van Rijn, H., & Borst, J. P. (2021). Discovering the brain stages of lexical decision: Behavioral effects originate from a single neural decision process. *Brain and Cognition*, *153*, 105786. https://doi.org/10.1016/j.bandc.2021.105786

Boehm, U., Van Maanen, L., Forstmann, B., & Van Rijn, H. (2014). Trial-by-trial fluctuations in CNV amplitude reflect anticipatory adjustment of response caution. *NeuroImage*, *96*, 95–105. https://doi.org/10.1016/j.neuroimage.2014.03.063

Borst, J. P., & Anderson, J. R. (2013). Using model-based functional MRI to locate working memory updates and declarative memory retrievals in the fronto-parietal network. *Proceedings of the National Academy of Sciences*, *110*(5), 1628–1633. https://doi.org/10.1073/pnas.1221572110

Borst, J. P., & Anderson, J. R. (2015). The discovery of processing stages: Analyzing EEG data with hidden semi-Markov models. *NeuroImage*, *108*, 60–73. https://doi.org/10.1016/j.neuroimage.2014.12.029

Borst, J. P., & Anderson, J. R. (2021). *Discovering Cognitive Stages in M/EEG Data to inform Cognitive Models*. Retrieved December 29, 2022, from http://jelmerborst.nl/pubs/ACTR_HsMM_MVPA_BorstAnderson_preprint.pdf

Borst, J. P., Schneider, D. W., Walsh, M. M., & Anderson, J. R. (2013). Stages of Processing in Associative Recognition: Evidence from Behavior, EEG, and Classification. *Journal of Cognitive Neuroscience*, *25*(12), 2151–2166. https://doi.org/10.1162/jocn_a_00457

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Broadbent, D. E. (1958). *Perception and communication*. Pergamon Press. https://doi.org/10.1037/10037-000

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014, September 2). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. arXiv: 1406.1078 [cs, stat]. Retrieved June 19, 2023, from http://arxiv.org/abs/1406.1078

Cohen, M. X. (2014, January 17). *Analyzing Neural Time Series Data: Theory and Practice* (1st edition). The MIT Press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Craik, A., He, Y., & Contreras-Vidal, J. L. (2019). Deep learning for electroencephalogram (EEG) classification tasks: A review. *Journal of Neural Engineering*, *16*(3), 031001. https://doi.org/10.1088/1741-2552/ab0ab5

Craik, F. I. M., & Lockhart, R. S. (1972). Levels of processing: A framework for memory research. *Journal of Verbal Learning and Verbal Behavior*, *11*(6), 671–684. https://doi.org/10.1016/S0022-5371(72)80001-X

Delorme, A., & Makeig, S. (2004). EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of Neuroscience Methods*, *134*(1), 9–21. https://doi.org/10.1016/j.jneumeth.2003.10.009

den Otter, R. (2024, January 24). *Hmp-ai*. Retrieved January 30, 2024, from https://github.com/rickdott/hmp-ai

Donders, F. C. (1868). On the speed of mental processes. *Acta Psychologica*, *30*, 412–431. https://doi.org/10.1016/0001-6918(69)90065-1

Dubarry, A.-S., Llorens, A., Trébuchon, A., Carron, R., Liégeois-Chauvel, C., Bénar, C.-G., & Alario, F.-X. (2017). Estimating parallel processing in a language task using single-trial intracerebral electroencephalography. *Psychological Science*, *28*(4), 414–426. https://doi.org/10.1177/0956797616681296

Enguehard, J. (2023, June 5). *Time Interpret: A Unified Model Interpretability Library for Time Series*. arXiv: 2306.02968 [cs]. https://doi.org/10.48550/arXiv.2306.02968

Gagnon-Audet, J.-C., Ahuja, K., Darvishi-Bayazi, M.-J., Mousavi, P., Dumas, G., & Rish, I. (2023, April 6). *WOODS: Benchmarks for Out-of-Distribution Generalization in Time Series*. arXiv: 2203.09978 [cs, stat]. Retrieved December 26, 2023, from http://arxiv.org/abs/2203.09978

Gramfort, A., Luessi, M., Larson, E., Engemann, D., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., & Hämäläinen, M. (2013). MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, *7*. Retrieved January 22, 2024, from https://www.frontiersin.org/articles/10.3389/fnins.2013.00267

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *06*(02), 107–116. https://doi.org/10.1142/S0218488598000094

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Höllig, J., Kulbach, C., & Thoma, S. (2023). TSInterpret: A python package for the interpretability of time series classification. *Journal of Open Source Software*, *8*(85), 5220. https://doi.org/10.21105/joss.05220

Ioffe, S., & Szegedy, C. (2015, March 2). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* (3). arXiv: 1502.03167 [cs]. Retrieved January 10, 2024, from http://arxiv.org/abs/1502.03167

Jas, M., Engemann, D. A., Bekhti, Y., Raimondo, F., & Gramfort, A. (2017). Autoreject: Automated artifact rejection for MEG and EEG data. *NeuroImage*, *159*, 417–429. https://doi.org/10.1016/j.neuroimage.2017.06.030

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., & Reblitz-Richardson, O. (2020). Captum: A unified and generic model interpretability library for PyTorch.

Kukačka, J., Golkov, V., & Cremers, D. (2017, October 29). *Regularization for Deep Learning: A Taxonomy*. arXiv: 1710.10686 [cs, stat]. Retrieved January 4, 2024, from http://arxiv.org/abs/1710.10686

Laird, J. E. (2019, August 20). *The Soar Cognitive Architecture*. MIT Press.

Lashgari, E., Liang, D., & Maoz, U. (2020). Data augmentation for deep-learning-based electroencephalography. *Journal of Neuroscience Methods*, *346*, 108885. https://doi.org/10.1016/j.jneumeth.2020.108885

LaValle, S. M., Branicky, M. S., & Lindemann, S. R. (2004). On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research*, *23*(7-8), 673–692.

Liu, G., Shih, K. J., Wang, T.-C., Reda, F. A., Sapra, K., Yu, Z., Tao, A., & Catanzaro, B. (2018, November 28). *Partial Convolution based Padding*. arXiv: 1811.11718 [cs]. Retrieved December 18, 2023, from http://arxiv.org/abs/1811.11718

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems 30* (pp. 4765–4774). Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Maaten, L. van der, & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*(86), 2579–2605. Retrieved January 28, 2024, from http://jmlr.org/papers/v9/vandermaaten08a.html

McClelland, J. L. (1979). On the time relations of mental processes: An examination of systems of processes in cascade. *Psychological Review*, *86*(4), 287–330. https://doi.org/10.1037/0033-295X.86.4.287

McClelland, J. L., Rumelhart, D. E., & Group, P. R. (1987, July 29). *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*. MIT Press.

McInnes, L., Healy, J., & Melville, J. (2020, September 17). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv: 1802.03426 [cs, stat]. https://doi.org/10.48550/arXiv.1802.03426

Mikolov, T., Kombrink, S., Burget, L., Černocký, J., & Khudanpur, S. (2011). Extensions of recurrent neural network language model. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5528–5531. https://doi.org/10.1109/ICASSP.2011.5947611

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*, 81–97. https://doi.org/10.1037/h0043158

Miller, G. A. (2003). The cognitive revolution: A historical perspective. *Trends in Cognitive Sciences*, *7*(3), 141–144. https://doi.org/10.1016/S1364-6613(03)00029-9

Molnar, C. (2022). *Interpretable machine learning. A Guide for Making Black Box Models Explainable* (2nd ed.). https://christophm.github.io/interpretable-ml-book

Müller, R., Kornblith, S., & Hinton, G. (2020, June 10). *When Does Label Smoothing Help?* arXiv: 1906.02629 [cs, stat]. https://doi.org/10.48550/arXiv.1906.02629

Newell, A. (1994, January 1). *Unified Theories of Cognition:* Harvard University Press.

Oberauer, K., & Kliegl, R. (2004). Simultaneous Cognitive Operations in Working Memory After Dual-Task Practice. *Journal of Experimental Psychology: Human Perception and Performance*, *30*(4), 689–707. https://doi.org/10.1037/0096-1523.30.4.689

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., . . . Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pohjonen, J., Stürenberg, C., Rannikko, A., Mirtti, T., & Pitkänen, E. (2022). Spectral decoupling allows training transferable neural networks in medical imaging. *iScience*, *25*(2), 103767. https://doi.org/10.1016/j.isci.2022.103767

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 1135–1144.

Santhakumaran, A. (2011, February 1). *Statistical Normalization and Back Propagation for Clasification*. https://doi.org/10.7763/IJCTE.2011.V3.288

Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Schomer, D. L., & Lopes da Silva, F. H. (2017, November). *Niedermeyer's electroencephalography: Basic principles, clinical applications, and related fields*. Oxford University Press. https://doi.org/10.1093/med/9780190228484.001.0001

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

Stanton, W. R. (2002). The dimensions of stage theories. *Current Psychology*, *21*(2), 176–198. https://doi.org/10.1007/s12144-002-1012-0

Sternberg, S. (1998). Discovering mental processing stages: The method of additive factors. In *Methods, models, and conceptual issues: An invitation to cognitive science, Vol. 4.* (pp. 703–863). The MIT Press.

Sundararajan, M., Taly, A., & Yan, Q. (2017, June 12). *Axiomatic Attribution for Deep Networks*. arXiv: 1703.01365 [cs]. Retrieved January 4, 2024, from http://arxiv.org/abs/1703.01365

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*, *27*. Retrieved June 19, 2023, from https://proceedings.neurips.cc/paper_files/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html

Van Maanen, L., & Van Rijn, H. (2010). The Locus of the Gratton Effect in Picture–Word Interference. *Topics in Cognitive Science*, *2*(1), 168–180. https://doi.org/10.1111/j.1756-8765.2009.01069.x

Van Maanen, L., Van Rijn, H., & Borst, J. P. (2009). Stroop and picture—word interference are two sides of the same coin. *Psychonomic Bulletin & Review*, *16*(6), 987–999. https://doi.org/10.3758/PBR.16.6.987

van Maanen, L., Portoles, O., & Borst, J. P. (2021). The Discovery and Interpretation of Evidence Accumulation Stages. *Computational Brain & Behavior*, *4*(4), 395–415. https://doi.org/10.1007/s42113-021-00105-2

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems*, *30*. Retrieved June 19, 2023, from https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Weindel, G. (2021, February 4). *On the measurement and estimation of cognitive processes with electrophysiological recordings and reaction time modeling*.

Weindel, G., gajdos, t., Burle, B., & Alario, P., Francois X. (2021, October). The decisive role of non-decision time for interpreting the parameters of decision making models. *PsyArXiv*. https://doi.org/10.31234/osf.io/gewb3

Weindel, G., Van Maanen, L., & Borst, J. P. (2024, January 29). *Trial-by-trial detection of cognitive events in neural time-series*.

Winkel, J., Hawkins, G. E., Ivry, R. B., Brown, S. D., Cools, R., & Forstmann, B. U. (2016). Focal striatum lesions impair cautiousness in humans. *Cortex*, *85*, 37–45. https://doi.org/10.1016/j.cortex.2016.09.023

Yu, S.-Z. (2010). Hidden semi-Markov models. *Artificial Intelligence*, *174*(2), 215–243. https://doi.org/10.1016/j.artint.2009.11.011

Zhang, H., Zhao, M., Wei, C., Mantini, D., Li, Z., & Liu, Q. (2021). EEGdenoiseNet: A benchmark dataset for deep learning solutions of EEG denoising. *Journal of Neural Engineering*, *18*(5), 056057. https://doi.org/10.1088/1741-2552/ac2bf8

Zhang, T., & Yu, B. (2005). Boosting with early stopping: Convergence and consistency. *The Annals of Statistics*, *33*(4). https://doi.org/10.1214/009053605000000255

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., . . . Wen, J.-R. (2023, May 7). *A Survey of Large Language Models*. arXiv: 2303.18223 [cs]. https://doi.org/10.48550/arXiv.2303.18223

Zylberberg, A., Dehaene, S., Roelfsema, P. R., & Sigman, M. (2011). The human Turing machine: A neural framework for mental programs. *Trends in Cognitive Sciences*, S136466131100088X. https://doi.org/10.1016/j.tics.2011.05.007

# A Model definitions

Models and model parameters can be further examined at (den Otter, 2024)

## A.1 Base

```
=====================================================================================
Layer (type)                              Output Shape          Param #
=====================================================================================
SAT1Base                                  [128, 5]              --
+ PartialConv2d                           [128, 64, 155, 30]    384
+ ReLU                                    [128, 64, 155, 30]    --
+ MaxPool2d                               [128, 64, 77, 30]     --
+ Conv2d                                  [128, 128, 75, 30]    24,704
+ ReLU                                    [128, 128, 75, 30]    --
+ MaxPool2d                               [128, 128, 37, 30]    --
+ Conv2d                                  [128, 256, 35, 30]    98,560
+ ReLU                                    [128, 256, 35, 30]    --
+ MaxPool2d                               [128, 256, 17, 30]    --
+ Flatten                                 [128, 130560]         --
+ Linear                                  [128, 128]            16,711,808
+ ReLU                                    [128, 128]            --
+ Dropout                                 [128, 128]            --
+ Linear                                  [128, 5]              645
=====================================================================================
Total params: 16,836,101
Trainable params: 16,836,101
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 22.73
=====================================================================================
Input size (MB): 2.44
Forward/backward pass size (MB): 875.04
Params size (MB): 67.34
Estimated Total Size (MB): 944.83
=====================================================================================
```

## A.2 Topological

```
=====================================================================================
Layer (type)                              Output Shape           Param #
=====================================================================================
SAT1Topological                           [128, 5]               --
+ PartialConv3d                           [128, 64, 795, 5, 8]   384
+ ReLU                                    [128, 64, 795, 5, 8]   --
+ MaxPool3d                               [128, 64, 397, 5, 8]   --
+ Conv3d                                  [128, 128, 395, 5, 8]  24,704
+ ReLU                                    [128, 128, 395, 5, 8]  --
+ MaxPool3d                               [128, 128, 197, 5, 8]  --
+ Conv3d                                  [128, 256, 195, 5, 8]  98,560
+ ReLU                                    [128, 256, 195, 5, 8]  --
+ MaxPool3d                               [128, 256, 97, 5, 8]   --
+ Flatten                                 [128, 993280]          --
+ Linear                                  [128, 128]             127,139,968
+ ReLU                                    [128, 128]             --
+ Dropout                                 [128, 128]             --
+ Linear                                  [128, 5]               645
=====================================================================================
Total params: 127,264,261
Trainable params: 127,264,261
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 166.20
=====================================================================================
Input size (MB): 16.36
Forward/backward pass size (MB): 6199.84
Params size (MB): 509.06
Estimated Total Size (MB): 6725.26
=====================================================================================
```

## A.3 Topological convolution

```
==========================================================================================
Layer (type)                              Output Shape              Param #
==========================================================================================
SAT1TopologicalConv                       [128, 5]                  --
+ PartialConv3d                           [128, 64, 795, 3, 6]      2,944
+ ReLU                                    [128, 64, 795, 3, 6]      --
+ MaxPool3d                               [128, 64, 397, 3, 6]      --
+ Conv3d                                  [128, 128, 395, 1, 4]     221,312
+ ReLU                                    [128, 128, 395, 1, 4]     --
+ MaxPool3d                               [128, 128, 197, 1, 4]     --
+ Conv3d                                  [128, 256, 195, 1, 4]     98,560
+ ReLU                                    [128, 256, 195, 1, 4]     --
+ MaxPool3d                               [128, 256, 97, 1, 4]      --
+ Flatten                                 [128, 99328]              --
+ Linear                                  [128, 128]                12,714,112
+ ReLU                                    [128, 128]                --
+ Dropout                                 [128, 128]                --
+ Linear                                  [128, 5]                  645
==========================================================================================
Total params: 13,037,573
Trainable params: 13,037,573
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 61.62
==========================================================================================
Input size (MB): 16.36
Forward/backward pass size (MB): 1349.52
Params size (MB): 52.15
Estimated Total Size (MB): 1418.04
==========================================================================================
```

## A.4 Deep

```
=====================================================================================
Layer (type)                              Output Shape          Param #
=====================================================================================
SAT1Deep                                  [128, 5]              --
+ PartialConv2d                           [128, 32, 775, 30]    832
+ ReLU                                    [128, 32, 775, 30]    --
+ MaxPool2d                               [128, 32, 387, 30]    --
+ Conv2d                                  [128, 64, 371, 30]    34,880
+ ReLU                                    [128, 64, 371, 30]    --
+ MaxPool2d                               [128, 64, 185, 30]    --
+ Conv2d                                  [128, 128, 175, 30]   90,240
+ ReLU                                    [128, 128, 175, 30]   --
+ MaxPool2d                               [128, 128, 87, 30]    --
+ Conv2d                                  [128, 256, 83, 30]    164,096
+ ReLU                                    [128, 256, 83, 30]    --
+ MaxPool2d                               [128, 256, 41, 30]    --
+ Conv2d                                  [128, 512, 39, 30]    393,728
+ ReLU                                    [128, 512, 39, 30]    --
+ MaxPool2d                               [128, 512, 19, 30]    --
+ Flatten                                 [128, 291840]         --
+ Linear                                  [128, 512]            149,422,592
+ ReLU                                    [128, 512]            --
+ Dropout                                 [128, 512]            --
+ Linear                                  [128, 5]              2,565
=====================================================================================
Total params: 150,108,933
Trainable params: 150,108,933
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 243.20
=====================================================================================
Input size (MB): 12.27
Forward/backward pass size (MB): 3446.08
Params size (MB): 600.44
Estimated Total Size (MB): 4058.79
=====================================================================================
```

## A.5 LSTM

```
=====================================================================================
Layer (type)                              Output Shape           Param #
=====================================================================================
SAT1LSTM                                  [128, 5]               --
+ LSTM                                    [102272, 256]          294,912
+ ReLU                                    [128, 799, 256]        --
+ Linear                                  [128, 799, 128]        32,896
+ Linear                                  [128, 799, 5]          645
=====================================================================================
Total params: 328,453
Trainable params: 328,453
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 7.72
=====================================================================================
Input size (MB): 12.27
Forward/backward pass size (MB): 318.27
Params size (MB): 1.31
Estimated Total Size (MB): 331.86
=====================================================================================
```

## A.6 GRU

```
==========================================================================================
Layer (type)                             Output Shape              Param #
==========================================================================================
SAT1GRU                                  [128, 5]                  --
+ GRU                                    [102272, 256]             221,184
+ ReLU                                   [128, 799, 256]           --
+ Linear                                 [128, 799, 128]           32,896
+ Linear                                 [128, 799, 5]             645
==========================================================================================
Total params: 254,725
Trainable params: 254,725
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 5.79
==========================================================================================
Input size (MB): 12.27
Forward/backward pass size (MB): 318.27
Params size (MB): 1.02
Estimated Total Size (MB): 331.56
==========================================================================================
```

## A.7 Transformer

```
=======================================================================================
Layer (type)                              Output Shape           Param #
=======================================================================================
TransformerModel                          [128, 5]               --
+ Linear                                  [128, 0, 30]           930
+ PositionalEncoding                      [128, 0, 30]           --
|    + Dropout                            [128, 0, 30]           --
+ TransformerEncoder                      [128, 0, 30]           --
|    + ModuleList                         --                     --
|    |    + TransformerEncoderLayer       [128, 0, 30]           35,102
|    |    + TransformerEncoderLayer       [128, 0, 30]           35,102
|    |    + TransformerEncoderLayer       [128, 0, 30]           35,102
|    |    + TransformerEncoderLayer       [128, 0, 30]           35,102
|    |    + TransformerEncoderLayer       [128, 0, 30]           35,102
|    |    + TransformerEncoderLayer       [128, 0, 30]           35,102
+ Linear                                  [128, 5]               155
=======================================================================================
Total params: 211,697
Trainable params: 211,697
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 24.24
=======================================================================================
Input size (MB): 2.47
Forward/backward pass size (MB): 0.01
Params size (MB): 0.76
Estimated Total Size (MB): 3.24
=======================================================================================
```

# B Additional figures

## B.1 Normalization (CNN)

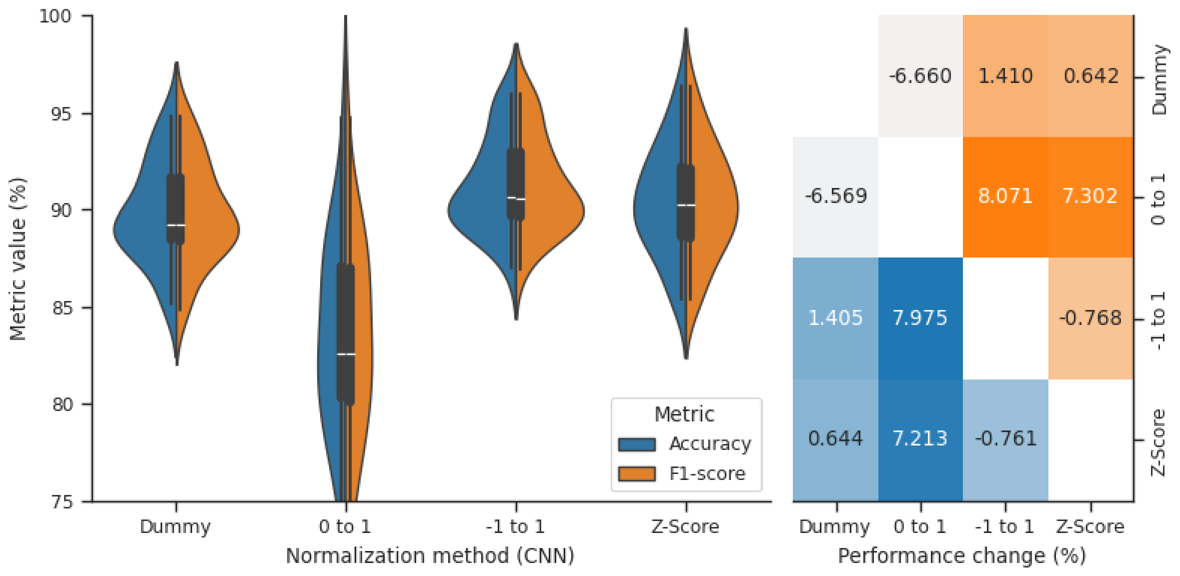

Figure 16: Results of normalization parameter test on CNN model. Normalizing to the range $[-1, 1]$ is optimal. Left panel: accuracy/F1-scores for each normalization method applied. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference between normalization methods.

| | Accuracy | F1-Score |
|---|---|---|
| Dummy | 89.86% (SD 2.54) | 89.84% (SD 2.57) |
| 0 to 1 | 83.29% (SD 5.10) | 83.18% (SD 5.23) |
| -1 to 1 | **91.26%** (SD 2.38) | **91.25%** (SD 2.39) |
| Z-Score | 90.50% (SD 2.79) | 90.48% (SD 2.80) |

Table 8: Performance metrics of normalization parameter test (CNN). Bolded numbers indicate the highest performance metrics.

## B.2 Performance (SAT1, 500 Hz)

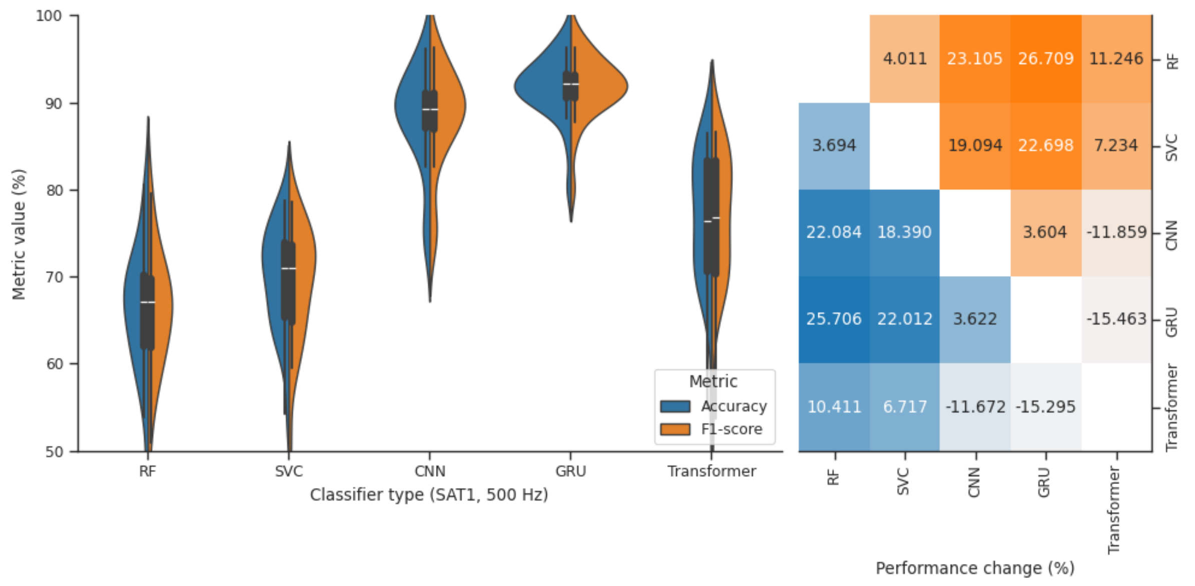

Figure 17: Results of performance test on SAT1, 500 Hz dataset.

## B.3 Generalization across conditions

### B.3.1 CNN

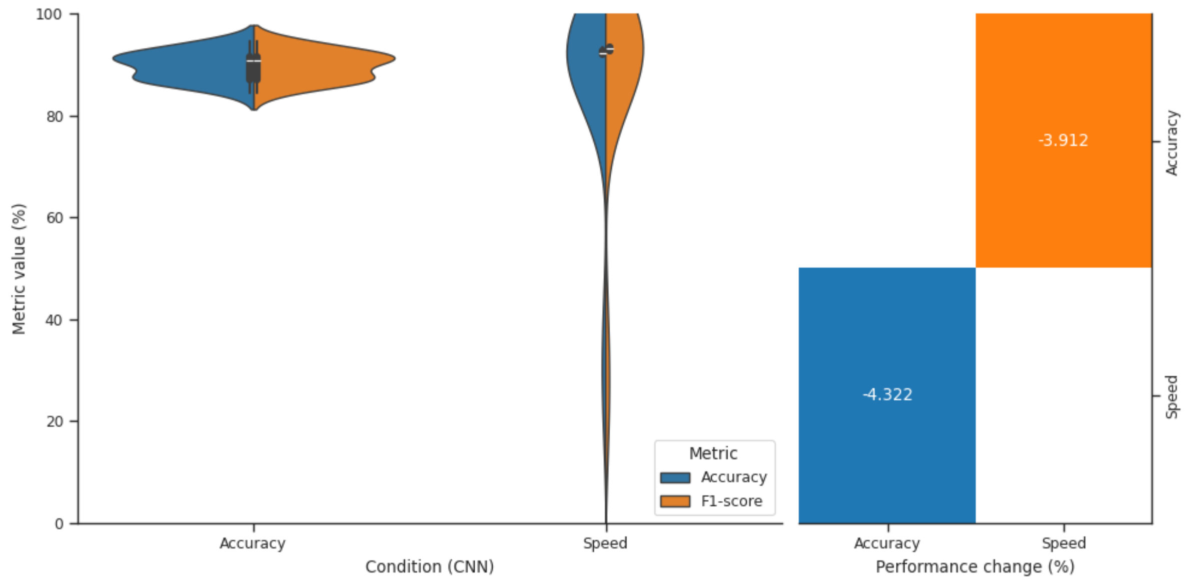

Figure 18: Generalization across conditions in a CNN model. Model trained and validated on accuracy condition, tested on speed. The CNN model is able to generalize across conditions. Left panel: accuracy/F1-scores for each condition. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference between conditions.

### B.3.2 Transformer

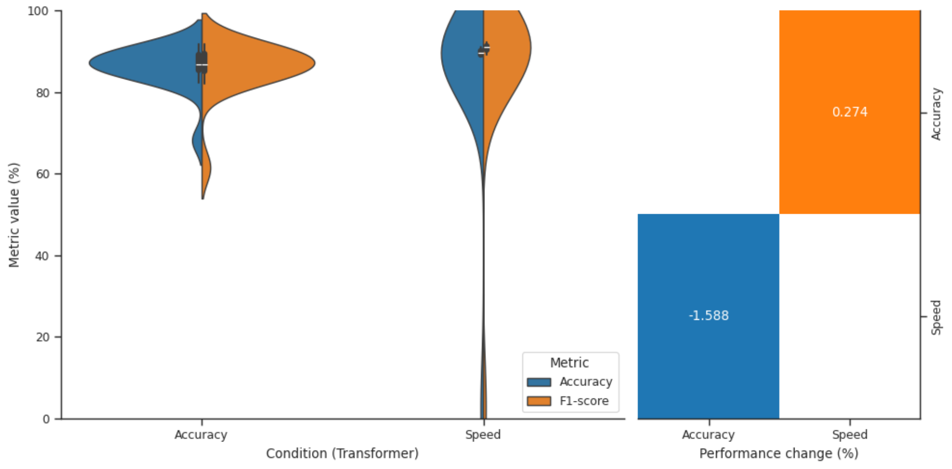

Figure 19: Generalization across conditions in a Transformer model. Model trained and validated on accuracy condition, tested on speed. The transformer model is able to generalize across conditions. Left panel: accuracy/F1-scores for each condition. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference between conditions.

## B.4 Generalization across labs & task

### B.4.1 CNN

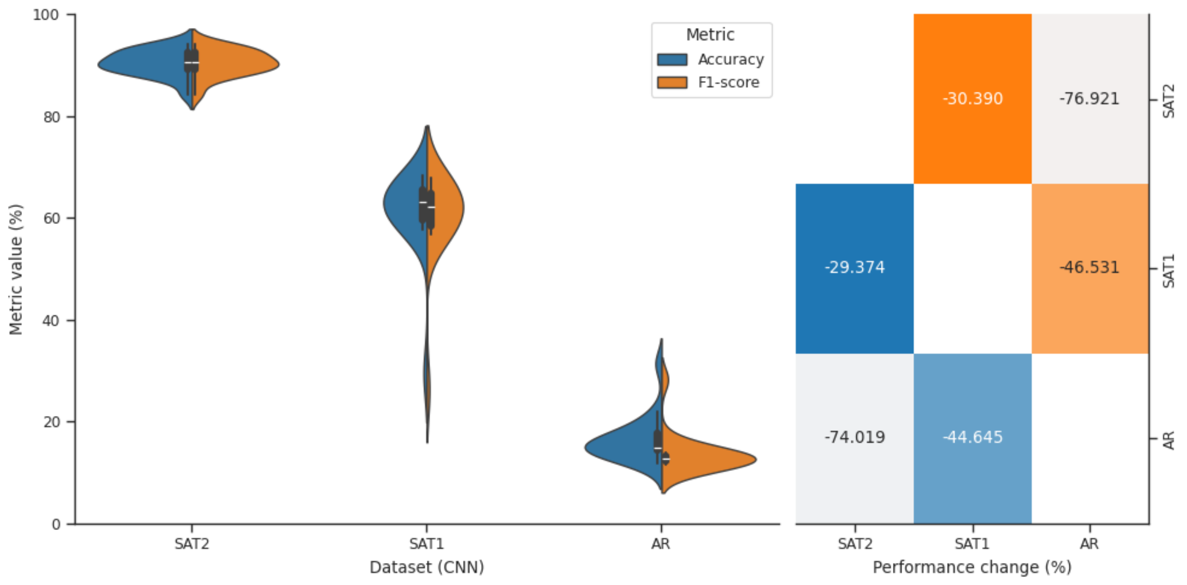

Figure 20: Generalization across labs & tasks in a CNN model. The CNN model is somewhat able to generalize across labs and not across tasks. Left panel: accuracy/F1-scores for each dataset. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference across datasets.

### B.4.2 Transformer

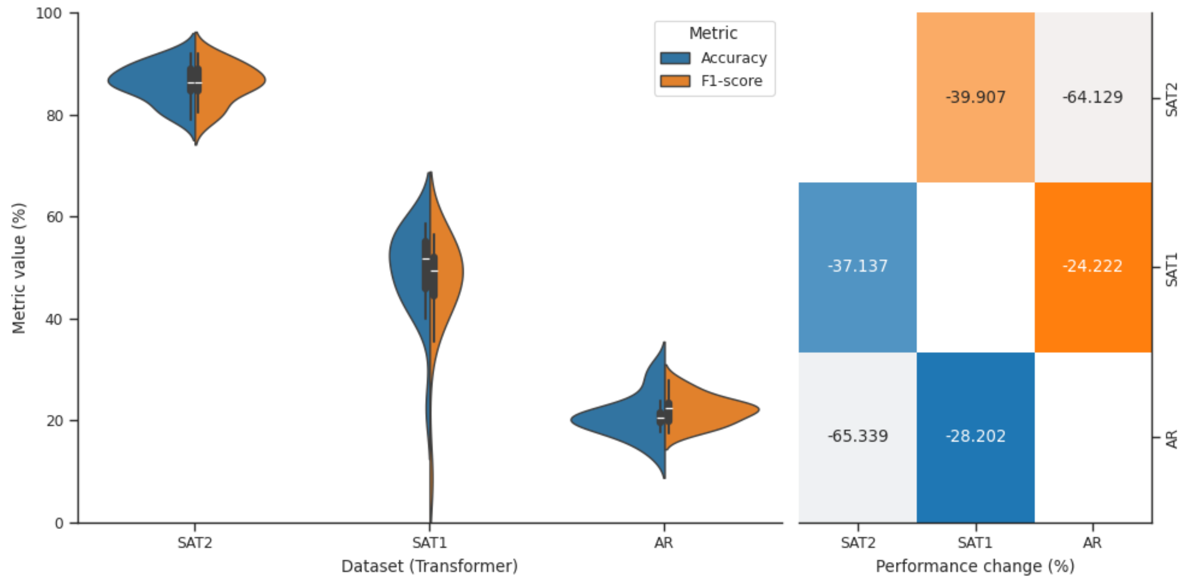

Figure 21: Generalization across labs & tasks in a Transformer model. The CNN model is somewhat able to generalize across labs and not across tasks. Left panel: accuracy/F1-scores for each dataset. Right panel: difference matrix (top half: F1-score (x - y), bottom half: accuracy (y - x)) displaying the pairwise difference across datasets.

# C Feature visualization

## C.1 GRU

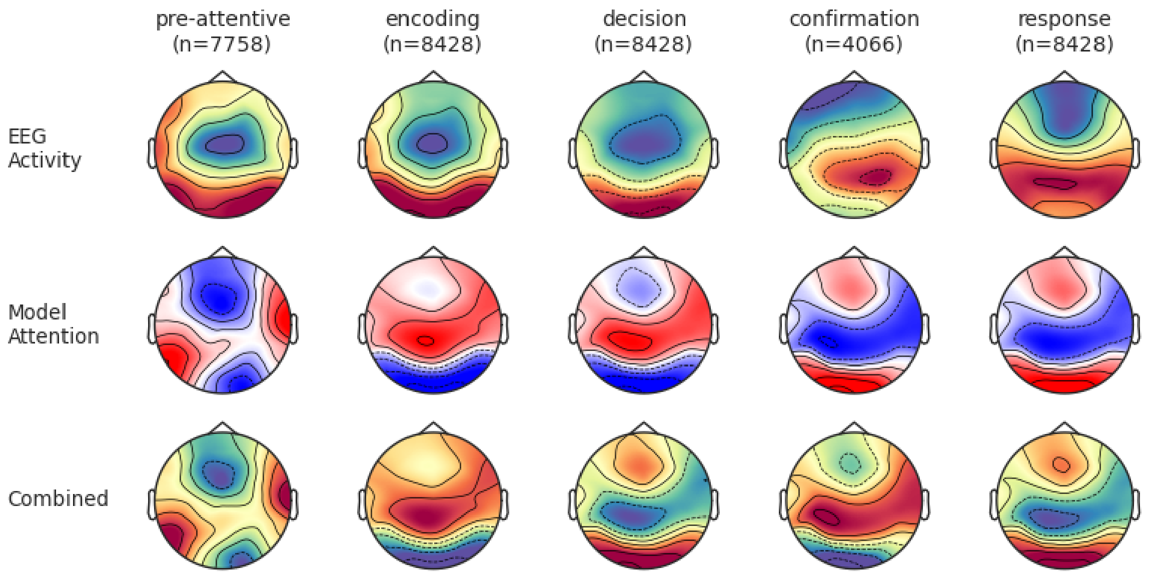

Figure 22: Integrated gradients visualization for the GRU model.

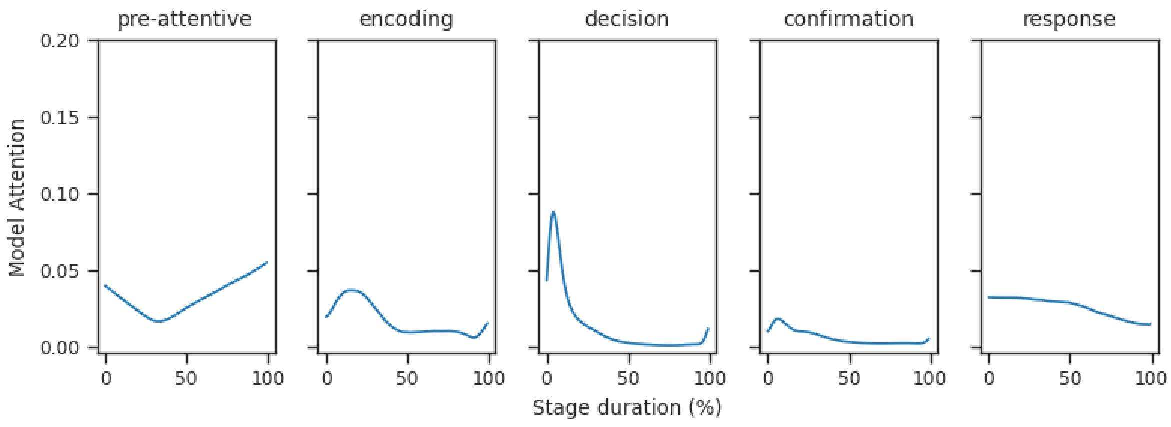

Figure 23: Interpolated model attention over stage duration for the GRU model.

## C.2 CNN

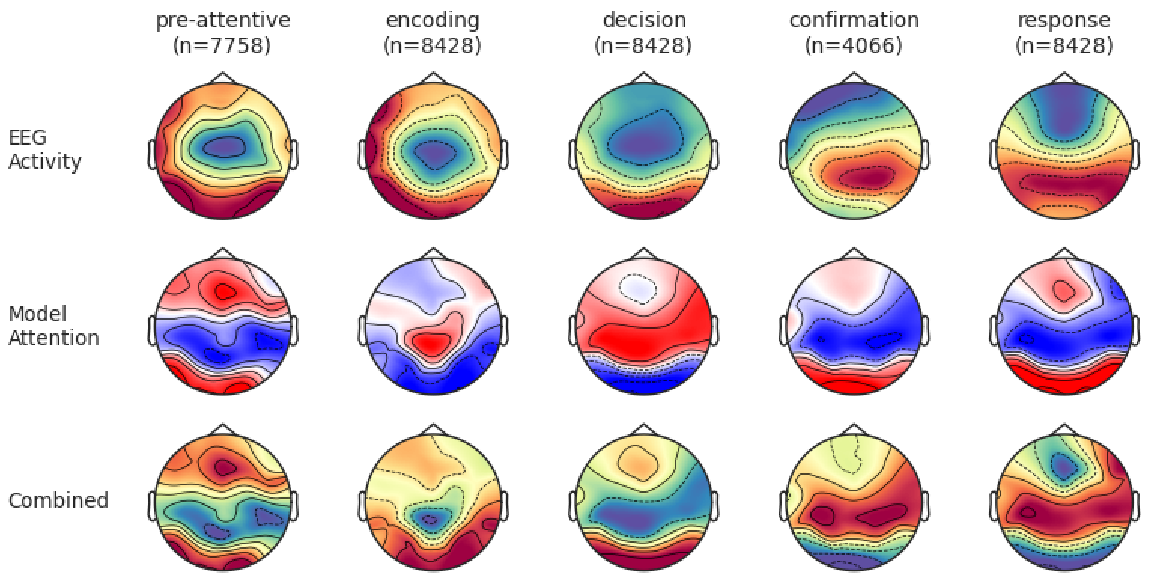

Figure 24: Integrated gradients visualization for the CNN model.

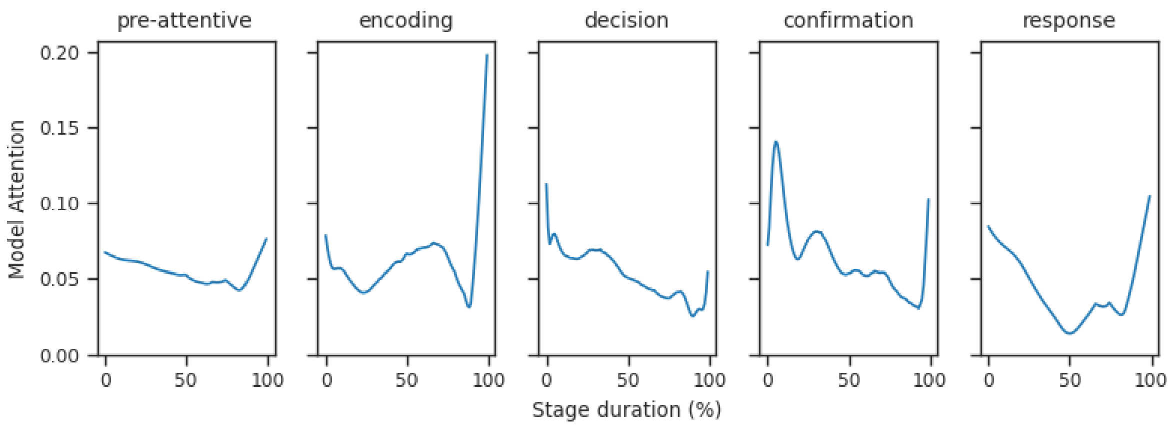

Figure 25: Interpolated model attention over stage duration for the CNN model.

## C.3 Transformer

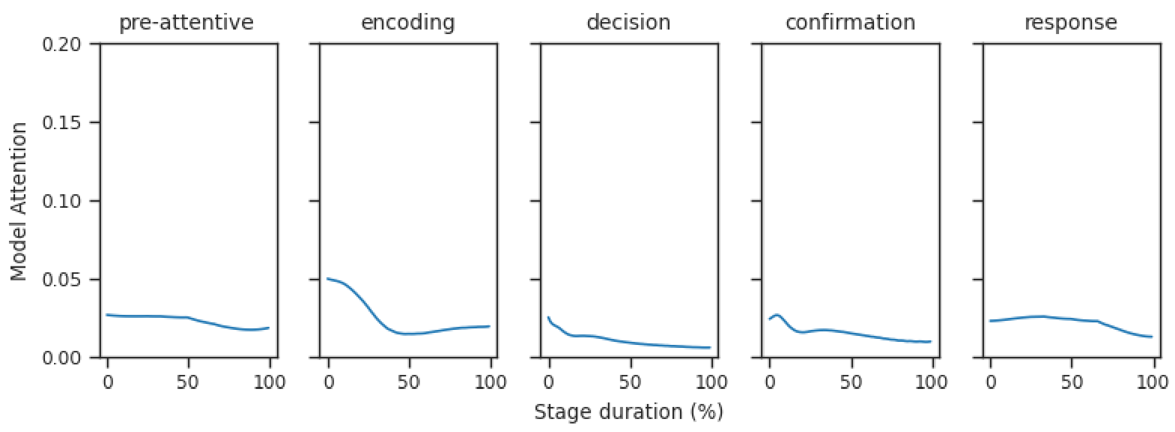

Figure 26: Interpolated model attention over stage duration for the transformer model.