Self-Supervised Learning Of Brain Dynamics From Broad Neuroimaging Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

Self-supervised learning techniques are celebrating immense success in natural language processing (NLP) by enabling models to learn from broad language data at unprecedented scales. Here, we aim to leverage the success of these techniques for mental state decoding, where researchers aim to identify specific mental states (such as an individual's experience of anger or happiness) from brain activity. To this end, we devise a set of novel self-supervised learning frameworks for neuroimaging data based on prominent learning frameworks in NLP. At their core, these frameworks learn the dynamics of brain activity by modeling sequences of activity akin to how NLP models sequences of text. We evaluate the performance of the proposed frameworks by pre-training models on a broad neuroimaging dataset spanning functional Magnetic Resonance Imaging (fMRI) data from $11,980$ experimental runs of 1726 individuals across 34 datasets and subsequently adapting the pre-trained models to two benchmark mental state decoding datasets. We show that the pre-trained models transfer well, outperforming baseline models when adapted to the data of only a few individuals, while models pre-trained in a learning framework based on causal language modeling clearly outperform the others.

1 Introduction

In mental state decoding ([1]), researchers aim to identify certain mental states (e.g., deciding to accept or reject a gamble or the experience of happiness or fear) from brain activity. To this end, researchers train predictive models to correctly identify (i.e., decode) these mental states from measured brain activity. At first sight, this approach seems straightforward, yet, researchers interested in training mental state decoding models are faced with the challenge that individual neuroimaging datasets are generally of high-dimension and small sample-size, such that individual samples often comprise many hundred thousand dimensions while each dataset includes only a few hundred samples for each of tens to hundreds of individuals. In this setting, predictive models are prone to overfitting, thereby not generalizing well to new individuals or datasets, which severely limits any generalizable insights that can be gained from training these models about the studied mental states and brain activity.

Neuroimaging research has recently entered a big data era, as individual researchers more frequently share their collected datasets publicly ([2]; [3]). In addition, several efforts have been made to standardize the data structure ([4]) and preprocessing ([5]) of neuroimaging data. Together, these developments open up new possibilities for the application pre-training in neuroimaging at scale, by training models on large and diverse sets of public neuroimaging data so that the resulting models can be adapted well to the datasets of individual researchers, without the need for much new data.

A wealth of empirical evidence has demonstrated that models pre-trained on large neuroimaging datasets generally achieve better mental state decoding performance on new data, while requiring less training time and data, when compared to models trained from scratch (e.g., [6]; [7]; [8]). Yet, much

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

of this work has either pre-trained models on large but homogenous datasets, such as data from many individuals who all perform the same few tasks at the same few acquisition sites (e.g., by using data from the Human Connectome Project; 9; 7) or trained models on highly-preprocessed input data (e.g., statistical maps summarizing the measured sequences of brain activity; 10), both limiting the downstream generalizability of the pre-trained models.

Here, we aim to overcome these limitations by leveraging recent advances in self-supervised learning to pre-train DL models on broad, minimally-preprocessed, and public neuroimaging data. Particularly, we take inspiration from natural language processing, where the application of self-supervised learning has led to unprecedented breakthroughs in the adaptive capabilities of models trained on public text data (e.g., 11; 12; 13; 14). Specifically, we test whether modeling sequences of brain activity akin to sequences of text enables the learning of generalizable representations of the dynamics of brain activity. To this end, we represent sequences of brain activity akin to the embedded representation of words in sequences of text, by describing each time point as a vector in which each dimension reflects the activity of a specific functional brain network at this point in time.

Contributions. We devise a set of novel self-supervised learning frameworks for neuroimaging data based on prominent learning frameworks in NLP, namely, sequence-to-sequence autoencoding (15), causal language modeling (11), and masked language modeling (12). We pre-train DL models with these frameworks on a large-scale, minimally-preprocessed upstream dataset comprising functional Magnetic Resonance Imaging (fMRI) data from $11,980$ experimental runs of 1726 individuals across 34 datasets. To our knowledge this represents one of the broadest datasets used for pre-training in neuroimaging to date, spanning a wealth of experimental conditions and domains as well as a diverse set of acquisition sites. We then evaluate the downstream adaptation performance of the pre-trained models in two benchmark mental state decoding datasets and show that models pre-trained in a causal sequence modeling framework clearly outperform the others, while all pre-trained models outperform a baseline model trained from scratch. To enable others to build on this work, we make our training data and pre-trained models publicly available (see Appendix A.4).

2 Methods overview

2.1 Input data: from brain volumes to brain networks

FMRI data are conventionally represented in four dimensions, such that the measured blood-oxygen-level-dependent (BOLD) signal is indicated in temporal sequences $S = \{V_1, ..., V_t\}$ of 3-dimensional volumes $V \in \mathbb{R}^{x \times y \times z}$ that show the BOLD signal intensity for each spatial location of the brain (as indicated by the three spatial dimensions x, y, and z). However, due to the strong spatial correlations of brain activity, the measured BOLD signal can also be summarized differently by representing it as a set $\Theta = \{\theta_1, ..., \theta_n\}$ of n brain networks θ, each describing the BOLD signal for some subset of voxels $v_{x,y,z} \subset V$ that exhibit correlated activity patterns.

Here, we utilize such a functional parcellation of brain activity to represent the measured BOLD signal, namely, Dictionaries of Functional Modes (DiFuMo), which were recently proposed by (16) and learned across millions of fMRI volumes. DiFuMo defines Θ through a sparse dictionary matrix $D \in \mathbb{R}^{p \times n}$ containing n networks, each defined by p values (one for each voxel $v \in V$ when flattened over the three spatial dimensions). The BOLD signal of volume V_t is then described as a linear combination of these networks, by the use of weights $\alpha_t \in \mathbb{R}^n$, such that $V_t = D\alpha_t$ and $\alpha_t = D^{\dagger}V_t$, where $D^{\dagger}$ indicates the pseudo-inverse of D: $D^{\dagger} = (D^T D)^{-1} D^T \in \mathbb{R}^{n \times p}$. We utilize a parcelation with $n = 1024$ networks, such that the resulting BOLD parcelation $X \in \mathbb{R}^{t \times n}$ describes the measured BOLD signal of each of these networks at each time point t of the input sequence S. Note that this representation of the input data is analogous to the embedded representation of sentences in natural language processing, where each word v is represented as some n-dimensional vector $v \in \mathbb{R}^n$.

2.2 Self-supervised learning frameworks: modeling sequences of brain activity akin to sequences of text

In the following, we describe our proposed set of self-supervised learning frameworks, which are based on prominent learning frameworks in NLP (for an overview, see Fig. 1). Before going into the details of each framework, we will establish several commonalities among them.

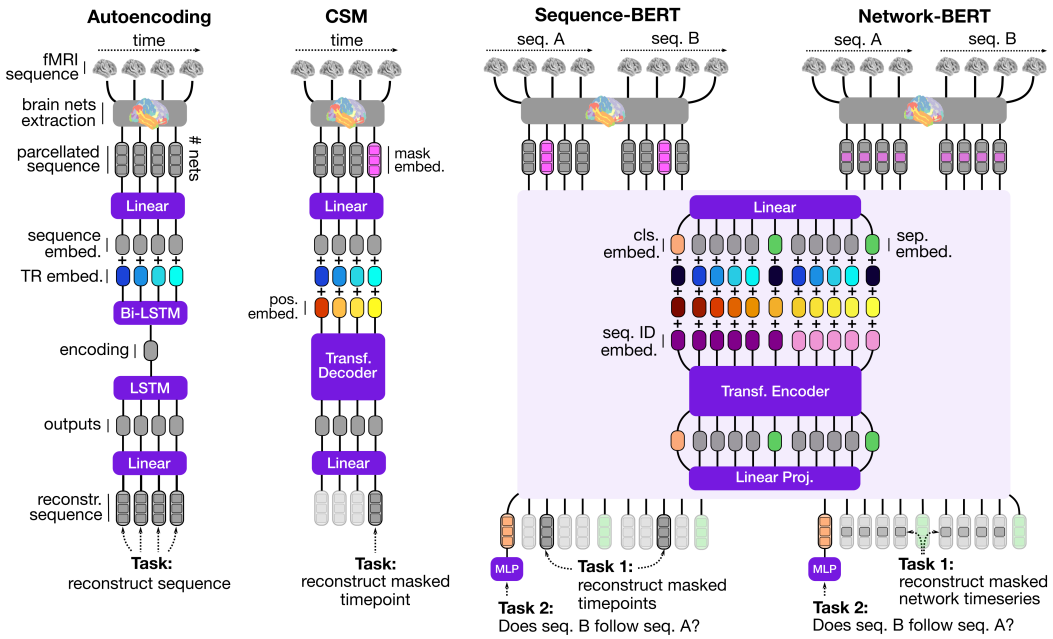

Figure 1: Proposed self-supervised learning frameworks for neuroimaging data, inspired by prominent learning frameworks in NLP.

2.2.1 Commonalities among frameworks

First, each framework takes as input a parcelated BOLD sequence $X \in \mathbb{R}^{t \times n}$ (with t time points for n networks) as well as the corresponding sampling frequency (i.e., repetition time or TR) of the underlying fMRI sequence (typically in the range of 0.7 s to 4 s).

Second, all frameworks linearly project the parcelated BOLD sequence X into an embedding representation $E^X \in \mathbb{R}^{t \times e}$: $E^X_{t,e} = b_e + \sum_n w_{n,e} X_{t,n}$ (where $w \in \mathbb{R}^{n \times e}$ and $b \in \mathbb{R}^e$ indicate weights and biases), thereby reducing the dimensionality of the input: $\mathbb{R}^{t \times n} \rightarrow \mathbb{R}^{t \times e}$, if $e < n$.

Third, to account for any differences in the sampling frequency of the fMRI input sequences, the frameworks add one of r learnable TR embeddings $E^{TR} \in \mathbb{R}^{r \times e}$ to each time point t in the BOLD embedding E^X. Each TR embedding thus corresponds to one possible time point k of the input. We set $r = 1500$ with a time increment of 0.2 s between any subsequent TR embeddings, such that $k \in 0, 0.2, ..., 300$ s. For example, for a sequence of 21 s with a TR of 0.7 s, we first round each time point $t \in 0, 0.7, ..., 21$ to the closest k and then add the corresponding E^{TR}_k to E^X_k. Note that we always label time points relative to the beginning of the input sequence, such that time point 0 is always assigned to the first sequence element.

Fourth, all frameworks utilize some standard DL model $f(\cdot)$ that takes E^{in} as input and predicts a corresponding output sequence $f(E^{in}) \in \mathbb{R}^{t \times e}$.

Fifth, during upstream learning, the output sequence $f(E^{in})$ is linearly projected back to the dimensionality of the parcelated BOLD sequence X, such that $\hat{X}_{t,n} = b_n + \sum_e w_{e,n} f(E^{in})_{t,e}$, where w and b again indicate weights and biases. This step is crucial, as all proposed learning frameworks involve reconstructing X (or parts of X) after some of its elements have been masked or it has been encoded in a lower-dimensional representation. To measure reconstruction performance, we utilize the mean absolute difference between the parcelated BOLD sequence X and the reconstructed BOLD sequence $\hat{X}$: $L_{rec} = \frac{1}{n(J)} \sum_{j \in J} |X_j - \hat{X}_j|$, where J indicates the set of elements that is to be reconstructed and $n(J)$ the number of elements in J.

Sixth, for downstream adaptation to a new mental state decoding task, the frameworks utilize a simple decoding head $p(\cdot)$, composed of a dense hidden layer with e model units (one for each embedding dimension, with $tanh$ activation) as well as a $softmax$ output layer with one model unit

i for each considered mental state in the data. $p(x)_i$ therefore indicates the probability that input x belongs to mental state i. Consequently, all frameworks treat the decoding task as a multinomial classification problem and use a standard cross entropy loss objective: $L_{cls} = -\sum_i y_i \log p(x)_i$, where y_i indicates a binary variable that equals 1 if i is the correct mental state and 0 otherwise.

2.2.2 Autoencoding

The autoencoding framework is inspired by recurrent sequence-to-sequence neural machine translation models (e.g., 15). These models take as input some text sequence, encode this sequence in a lower-dimensional representation, and predict an output sequence from the lower-dimensional encoding (e.g., a translation of the original input text to another language).

Accordingly, our autoencoding framework uses a standard recurrent autoencoder architecture $f_a(\cdot)$, with an encoder and decoder part, each comprised of a stack of long short-term memory (LSTM; 17) units (see Fig. 1). The encoder $e(\cdot)$ takes E^{in} as input and encodes it in a lower-dimensional representation $h \in \mathbb{R}^e$. The decoder $d(\cdot)$ then takes h as input and predicts a corresponding output sequence $d(h) \in \mathbb{R}^{t \times e}$, such that $f_a(E^{in}) = d(h)$. Our encoder is composed of a stack of bidirectional LSTM units, whereas the decoder is build from a stack of unidirectional LSTM units. Note that we apply residual connections between the LSTM units in a stack: Let $LSTM^i$ and $LSTM^{(i+1)}$ be the i-th and $(i+1)$-th LSTM layer in a stack (with weights w^i and w^{i+1}). At the t-th step of an input sequence, the input and outputs of layers i and $(i+1)$ are defined as: $h_t^i, c_t^i = LSTM^i(h_{t-1}^i, c_{t-1}^i, x_{t-1}^{i-1}; w^i)$; $x_t^i = h_t^i + x_{t-1}^{i-1}$; $h_{t+1}^{i+1}, c_{t+1}^{i+1} = LSTM^{i+1}(h_t^{i+1}, c_t^{i+1}, x_t^i; w^{i+1})$, where $x_t^i \in \mathbb{R}^e$ represents the input to $LSTM^i$ at step t, while $h_t^i \in \mathbb{R}^e$ and $c_t^i \in \mathbb{R}^e$ represent the $LSTM$'s hidden and cell states respectively. For the encoder, we average the hidden states $h_t^{z \in i}$ and $h_t^{y \in i}$ of the two LSTM units z and y contained in any bidirectional LSTM layer i at each time point t: $h_t^i = 0.5(h_t^{z \in i} + h_t^{y \in i})$. The encoded sequence embedding h that is forwarded to the decoder is then defined as the mean of the final hidden states of the two LSTM units contained in the encoder's final layer.

Upstream learning. During pre-training, we jointly train the encoder and decoder to reconstruct the parcelated BOLD sequence X by minimizing its mean absolute difference to the reconstructed BOLD sequence $\hat{X}$ (see section 2.2.1): $\min \frac{1}{t} \sum_{i=1}^{t} \frac{1}{e} \sum_{j=1}^{e} |\hat{X}_{i,j} - X_{i,j}|$. Importantly, we apply teacher-forcing (18) during upstream learning by using E^{in} in 50% of the cases as each next input for the decoder, instead of using the decoder's own preceding output.

Adaptation. To adapt the autoencoding framework to a new decoding task, we forward h as input to a corresponding decoding head $p(\cdot)$ (see section 2.2.1), thereby omitting the decoder $d(\cdot)$.

2.2.3 Causal sequence modeling (CSM)

The causal sequence modeling (CSM) framework is inspired by recent advances in causal language modeling (11), where models are trained to predict the next word in a sequence of text. Specifically, these models receive some text sequence as input, e.g., represented as a matrix $X \in \mathbb{R}^{n \times e}$ with n words each encoded by an e-dimensional vector, where the last sequence element (representing the last word) has been replaced by some some learned mask embedding $E^{msk} \in \mathbb{R}^e$ (e.g., "When it rains, it [msk]"). The task of a model is then to predict the word that was replaced by E^{msk}. In causal language modeling, models are therefore solely concerned with the part of the input sequence that precedes the masked word.

To adapt causal language modeling to neuroimaging data, we mask the last time point T of the parcelated BOLD sequence X by replacing it with a learnable mask embedding E^{msk}. As for the autoencoding framework, X is then transformed to E^{in} by means of linear projection and the addition of TR embeddings E^{TR}. In line with recent advances in causal language modeling, we utilize a standard transformer decoder model $f_d(\cdot)$ (based on the GPT architecture; 11) in this learning framework. As transformer models are not aware of the ordering of their input, we further add learnable position embeddings $E^{pos} \in \mathbb{R}^{p \times e}$ to E^{in}, each representing one of p possible positions in the input sequence (we set $p = 512$ for all transformer models). Given E^{in}, the transformer decoder model predicts a corresponding output sequence $f_d(E^{in}) \in \mathbb{R}^{t \times e}$ in the form of the hidden states at the output of its last layer.

Upstream learning. In line with causal language modeling, the upstream objective of the CSM framework is to reconstruct the masked last time point T of the parcelated BOLD sequence X by minimizing the mean absolute difference to its reconstruction $\hat{X}_T$: $\min \frac{1}{e} \sum_{i=1}^{e} |\hat{X}_{T,i} - X_{T,i}|$

Adaptation. During downstream adaptation, we attach a learnable classification embedding $E^{cls} \in \mathbb{R}^e$ to the end of BOLD embedding E^X and forward the resulting prediction $f(E^{in})_{T+1}$ to a corresponding decoding head $p(\cdot)$ (see section 2.2.1). Note that we use the same learnable TR embedding for all embeddings that are attached to an input sequence during training.

2.2.4 Sequence-BERT

The Sequence-BERT framework is based on recent advances in bidirectional masked language modeling, particularly BERT (12). In NLP, BERT is trained to jointly solve a masked-language-modeling and next-sentence-prediction task. To this end, BERT receives two masked sentences S^a (e.g., "The sky is [msk].") and S^b (e.g., "The [msk] is shining.") as input, in which some fraction of words have been randomly masked. BERT is then asked to I) predict the masked words (masked-language-modeling) and II) determine whether S^b follows S^a (next-sentence-prediction). In contrast to causal language modeling, masked language modeling is considered as a bidirectional learning task, as the model is not just concerned with the parts of the input that precede a masked word but also the parts that follow it.

To adapt the BERT framework to neuroimaging data, we first sample two input sequences $X^a \in \mathbb{R}^{t^a \times n}$ and $X^b \in \mathbb{R}^{t^b \times n}$ from the data (with lengths t^a and t^b). In 50% of the cases, the two sequences represent sequential parts of the same underlying fMRI sequence, while in the other 50% of cases, the two sequences are randomly sampled from two distinct fMRI sequences. Due to the strong temporal autocorrelation of BOLD data, we randomly leave a gap of 1-5 TRs between the two sequences when sampling them from the same underlying fMRI sequence, to encourage the models to consider the entire input sequences and not just their connecting time points. As for BERT, we next randomly mask time points of each sampled sequence (with a probability of 20%), by replacing them with a learnable mask embedding E^{msk}. The masked versions of X^a and X^b are then linearly projected to obtain respective BOLD embeddings E^{X^a} and E^{X^b} to which we then add TR and position embeddings E^{TR} and E^{pos} as well as a sequence ID embedding $E^{id} \in \mathbb{R}^{2 \times e}$ (to indicate each of the two input sequences). In line with BERT, we lastly also attach a separation embedding $E^{sep} \in \mathbb{R}^e$ to the end of each sequence (to indicate their endpoints) and a classification embedding E^{cls} to the beginning of S^a (for the next-sentence-prediction task; see upstream learning below). The resulting embedding representation $E^{in} = [E^{cls}, \{E_t^{X^a} + E_t^{TR} + E_t^{pos} + E_1^{id}\}^{t \in t^a}, E^{sep}, \{E_t^{X^b} + E_t^{TR} + E_t^{pos} + E_2^{id}\}^{t \in t^b}, E^{sep}]$ is then forwarded to a standard transformer encoder model $f_e(\cdot)$ (based on the BERT architecture; 12), which outputs a corresponding output sequence $f_e(E^{in}) \in \mathbb{R}^{t \times e}$ in the form of the hidden states at the output of its final layer.

Upstream learning. Similar to BERT, we define the upstream objective of the Sequence-BERT framework as the sum of two learning objectives: In line with masked-language-modeling, the first objective is defined as the reconstruction loss L_{rec} for all masked time points J: $L_{rec} = \frac{1}{n(J)} \sum_{j \in J} \frac{1}{e} \sum_{i=1}^{e} |X_{j,i} - \hat{X}_{j,i}|$. In line with next-sentence-prediction, the other objective is defined as the binary cross entropy loss for a decoding head that receives $f(E^{in})_1$ (corresponding to the transformer output for classification embedding E^{cls}) as input and predicts probability $p(f(E^{in})_1)$ that X^b follows X^a: $L_{cls} = -y \log(p(f(E^{in})_1)) - (1-y) \log(1 - p(f(E^{in})_1))$, where y indicates a binary variable that equals 1 if the two sequences come from the same fMRI run and 0 otherwise.

Adaptation. During downstream adaptation, we forward the transformer's output for classification embedding $f_e(E^{in})_1$ as input to a corresponding decoding head $p(\cdot)$ (see section 2.2.1).

2.2.5 Network-BERT

All so far presented frameworks model their input on the level of the individual time points in a sequence, by trying to reconstruct the parcelated whole-brain BOLD signal for specific time points. Yet, the dynamics of brain activity data can also be viewed differently, by focusing on the interaction of network activities over time instead of focusing on the distribution of network activities at individual

time points. We therefore created a variant of the Sequence-BERT framework that randomly (at a rate of 10%) masks the activity of individual networks n in the parcelated BOLD sequence X by replacing them with a learnable network mask embedding $E^{msk} \in \mathbb{R}^t$, comprising one value for each possible time point in the input (see Fig. 1). With the exception of the masking procedure, the Network-BERT and Sequence-BERT frameworks are identical.

Upstream learning. During upstream learning, Network-BERT uses the same cross-entropy loss L_{cls} as Sequence-BERT and adds it to a variant of the reconstruction loss that measures reconstruction performance for masked network time courses J: $L_{rec} = \frac{1}{n(J)} \sum_{j \in J} \frac{1}{t} \sum_{i=1}^{t} |X_{i,j} - \hat{X}_{i,j}|$.

Adaptation. During downstream adaptation, Network-BERT forwards the transformer's output for the classification embedding $f_e(E^{in})_1$ to a respective decoding head $p(\cdot)$ (see section 2.2.1).

2.3 Training details

If not reported otherwise, we train all models with stochastic gradient descent and the ADAM optimizer (with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$; (19)). We further use a linear learning rate decay schedule (with a warmup phase of 1% of the total number of training steps), gradient norm clipping at 1.0, and $L2$-regularisation (weighted by 0.1).

During upstream learning, we set the maximum learning rates to $2e^{-4}$, $5e^{-4}$, $1e^{-4}$, and $1e^{-4}$ for the autoencoding, CSM, Sequence-BERT, and Network-BERT frameworks respectively (based on the learning rates used in corresponding NLP studies; 11; 12; 15) and randomly sample input sequences of 10 to 55 TRs from the fMRI files contained in our upstream dataset. For the two BERT frameworks, each sequence S_i is split into two sequences S_i^a and S_i^b of equal length (with a randomly sampled gap of 1 to 5 TRs between the two sequences), while S_i^b is then randomly exchanged (with 50% probability) with another S_j^b (where $i \neq j$). For each upstream training run, we then randomly divide the upstream data into distinct training and validation datasets by randomly designating $\%5$ of the fMRI run files of each upstream dataset as validation data (at a minimum of 2 files per dataset) and using the rest of the files for training. At each evaluation step, we randomly sample $640,000$ from the validation dataset to evaluate model performances.

During downstream adaptation, we begin training with the respective pre-trained model parameters and then allow all parameters to be freely changed during the adaptation. Other optimization and architecture hyper-parameters are evaluated and reported in the experiments described below.

Compute resources. Upstream learning was performed on Google Compute Engine n1-highmem-64 nodes with four Nvidia Tesla P100 GPUs, 64 CPU threads, and 416 GB RAM memory, while downstream adaptation was performed on compute nodes of the Texas Advanced Computing Center with one Nvidia 1080-TI GPU, 32 CPU threads, and 128 GB RAM memory. Training times varied depending on model size but were generally in the range of 1 to 3 days for upstream learning and 0.5 to 3 hours for downstream adaptations.

3 Experiments

3.1 Datasets

All unprocessed fMRI data used in this study are publicly available through OpenNeuro.org (3) and the Human Connectome Project (HCP; 20). For an overview of our preprocessing, see Appendix A.3.

3.1.1 Upstream: $11,980$ fMRI runs from 1726 individuals across 34 datasets

Our upstream dataset comprises $11,980$ fMRI run files from 1726 individuals across 34 datasets (for an overview, see Fig. 2 and Appendix Table A.1). To our knowledge, this represents one of the broadest fMRI datasets used for pre-training in neuroimaging to date, spanning many acquisition sites and a diverse set of experimental conditions and domains. A few prominent examples of included datasets are The Midnight Scan Club (21), Individual Brain Charting (22), Amsterdam Open MRI collection (23), BOLD5000 (24), and the Narratives collection (25).

3.1.2 Downstream: 2 benchmark datasets with 20 and 26 mental states

To evaluate downstream adaptation performance, we use two benchmark mental state decoding datasets, which span a wide range of mental states, namely, task-fMRI data from the HCP ([20]) and the Multi-Domain Task Battery (MDTB; [26]). Our HCP dataset includes data from 100 individuals performing two fMRI runs for each of seven tasks, with a total of 19 distinct mental states across the tasks (see Appendix A.2). We also include two resting state fMRI runs per individual, in which individuals simply rest in the fMRI without any particular task, thereby increasing the total number of mental states to 20 for the HCP data. The MDTB data include 24 individuals performing 26 tasks in two scanning sessions with eight fMRI runs per session. Due to the strong similarity of experimental conditions within individual tasks of the MDTB, we defined each of its tasks as one target mental state (see Appendix A.2). During downstream adaptation, we sample sequences from both datasets according to their experiment event files, which define an on- and offset for each trial in an fMRI run (with durations generally ranging between 2 and 30 s, at a TR of 0.72 (HCP) and 1 s (MDTB)). For the HCP's resting state data, we randomly sampled sequences between 10 and 30 s.

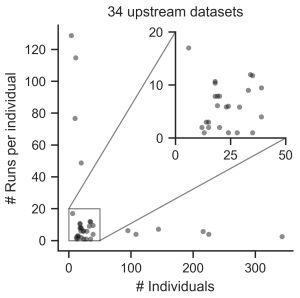

Figure 2: Number of individuals and average number of fMRI runs per individual in our upstream data.

3.2 Hyper-parameter evaluation: larger embeddings but not deeper models

In our first experiment, we compare the performance of different hyper-parameter settings for the proposed frameworks. Specifically, we evaluate three different embedding dimensions e (192, 384, and 768; see section 2.2.1) and three different numbers of hidden layers for the used DL models (2, 4, and 6 hidden layers for the encoder and decoder parts of the autoencoding framework and 4, 8, and 12 hidden layers for the transformer models). In line with other work ([27]), we scale the number of attention heads n_α per layer of the transformer models with the embedding dimension, such that $n_\alpha = \frac{e}{64}$. We then train one model for each point of the resulting 9-point grid of each framework for $200,000$ training steps with a mini-batch size of 256 random samples per step.

Overall, model performances improve (in terms of validation loss) with larger embedding dimensions but not with the number of hidden layers (see Fig. 3). To this end, we decided to continue all further analyses with models comprising 768 embedding dimensions and 2 hidden layers for the encoder and decoder parts of the autoencoding framework and 4 hidden layers for the transformer models.

3.3 Upstream performance: models learn well in all frameworks

In experiment two, we then fully pre-trained one model in each framework according to our previously selected set of hyper-parameters (see section 3.2), by training models for $300,000$ training steps at a mini-batch size of 768. All models learned well over the course of their training, with gradually decreasing training and validation losses (see Fig. 4). Note that the transformer model trained in CSM slightly overfitted the upstream data while the gap in training and validation performance of the autoencoding framework results from our application of teacher-forcing during training (see section 2.2.2). We also evaluated the reconstruction performance of the final models for different parts of the brain (see Appendix A.6) and found that the frameworks exhibited similar overall distributions of reconstruction error, with relatively higher errors in the posterior parietal, occipital, and cingulate cortices as well parts of the limbic system.

3.4 Adapting pre-trained language models does not yield meaningful upstream performance gains

In our third experiment, we tested whether adapting pre-trained language models to our upstream data would yield meaningful performance gains, when compared to the models trained from scratch. To this end, we adapted pre-trained variants of GPT-2 ([28]) and BERT ([12]) (with 12 hidden layers and an embedding dimension of 768; as provided by Hugging Face's Transformer library ([29])) to our upstream dataset (see Appendix A.7). However, the upstream validation performance of these

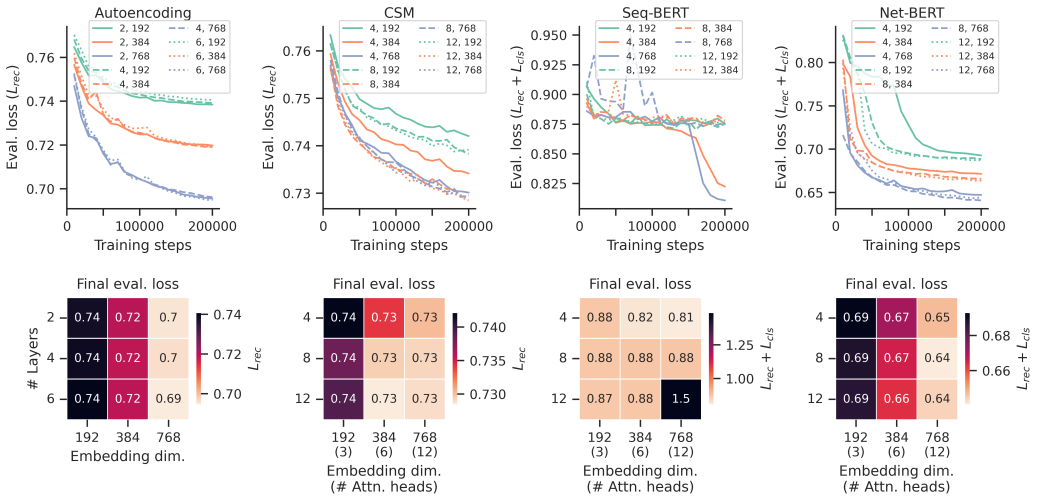

Figure 3: Hyper-parameter evaluation: model performances generally improve with larger embedding dimensions but not with model depth. Note that we do not plot the evaluation loss for the Sequence-BERT variant with 12 hidden layers and 768 embedding dimensions, as the loss of this model variant did not meaningfully change over the course of its training (see Appendix A.5).

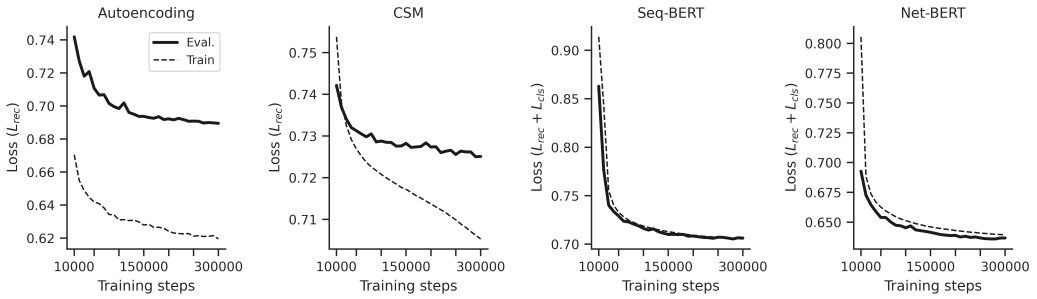

Figure 4: Upstream learning performance of each framework with the selected autoencoder and transformer model architecture configurations (see section 3.2).

models did not meaningfully improve upon the performance of our models trained from scratch, at final upstream validation losses of 0.73 (CSM with GPT-2) and 0.80 (Sequence-BERT with BERT).

3.5 Downstream adaptation: CSM outperforms other frameworks

Lastly, in our fourth experiment, we evaluated the downstream adaptation performance of our pre-trained models in two benchmark mental state decoding datasets (see section 3.1.2). To gauge the effectiveness of pre-training, we adapted the pre-trained models to varying sizes of the downstream datasets, comprising the data from 1 to 48 (HCP) and 11 (MDTB) individuals. Importantly, for each training run, we randomly split the data into training, validation, and test datasets, by randomly assigning individuals to one of these three datasets. Specifically, for each run, we first randomly selected 10 (HCP) and 3 (MDTB) individuals whose data we use for validation and 20 (HCP) and 9 (MDTB) other individuals whose data we use for testing, before randomly sampling the given number of individuals whose data we use for training from the remaining pool of individuals. We then adapt the pre-trained models to the training data at a mini-batch size of 256 with varying learning rates and numbers of training steps (see Appendix A.9 for an overview of our downstream hyper-parameter evaluation). Here, we only report final test decoding accuracies for those models that achieved the highest final validation decoding accuracy (see Appendix Fig. A4 and A5 for individual validation

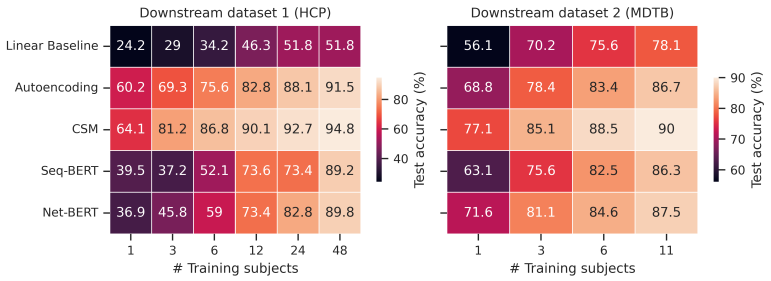

Figure 5: Downstream adaptation performance of our pre-trained models in two benchmark mental state decoding datasets. We adapted models to the data of varying numbers of individuals, while always measuring final model decoding accuracies on a test dataset comprising 20 and 9 left-out individuals from the HCP and MDTB datasets respectively.

accuracies). In line with the state-of-the-art in mental state decoding (see [30]), we also evaluated the performance of a linear baseline model on the both datasets, which we trained from scratch each time (see Appendix A.8).

The pre-trained models clearly outperformed the linear baseline model in their test decoding accuracy across the different sizes of the training dataset (at chance-levels of 5.0% (HCP) and 3.9% (MDTB)), while the models' performances also scaled well with the number of individuals in the training data (Fig. 5). For the HCP data, for which other reported decoding performances exist, our pre-trained models perform on par with models trained on much larger fractions of the data (often comprising the data of many hundred individuals; [9; 7; 6; 31]), which generally report test decoding accuracies in the range of 80 to 93%. Overall, the transformer decoder model trained with the CSM framework clearly outperformed the other models by achieving the highest test decoding accuracy throughout all evaluated sizes of the training dataset.

Note that model performances were very stable across the different random splits of the data (see Fig. 5 and Appendix Fig. A4 and A4), strongly suggesting stability in the models' performance towards different possible splits of the data. In addition, we tested whether model performances are also stable over other non-deterministic aspects of the training by replicating our downstream adaptation analysis for the MDTB dataset with a different set of random seeds. Again, model performances were very stable across the different random seeds, yielding almost identical results in the replication when compared to our original analysis (see Appendix A.10).

4 Conclusion

We propose a set of novel self-supervised learning frameworks for neuroimaging data based on prominent counterparts in natural language processing. By the use of these frameworks, we are able to pre-train DL models across broad, public neuroimaging data, comprising many individuals, experimental domains, and acquisition sites. We show that the pre-trained models generalize well to two benchmark neuroimaging datasets in a mental state decoding analysis, outperforming baseline models even when trained on small subsets of the data. To enable others to build on this work, we make our training data and pre-trained models publicly available (see Appendix A.4).

Limitations and potential negative social impact. While the proposed frameworks perform well in training models that generalize to new neuroimaging datasets, this work currently neglects another key aspect of mental state decoding, namely, the ability to draw inferences about the association between decoded mental states and brain activity from the trained models. First empirical evidence indicates that attribution methods from explainable artificial intelligence (XAI; [32]) research are well-suited to provide insights in the mental state decoding decisions of DL models (for a detailed discussion, see [33]). Yet, further research is needed to understand their limits in mental state decoding.

We are currently not aware of any direct negative social impacts that could follow from pre-training DL models on public neuroimaging data, which are de-identified and collected and shared with human consent in a way approved by respective institutional review boards.

References

[1] J.-D. Haynes and G. Rees, "Decoding mental states from brain activity in humans," *Nature Reviews Neuroscience*, vol. 7, no. 7, pp. 523–534, 2006.

[2] C. Horien, S. Noble, A. S. Greene, K. Lee, D. S. Barron, S. Gao, D. O'Connor, M. Salehi, J. Dadashkarimi, X. Shen, *et al.*, "A hitchhiker's guide to working with large, open-source neuroimaging datasets," *Nature human behaviour*, vol. 5, no. 2, pp. 185–193, 2021.

[3] C. J. Markiewicz, K. J. Gorgolewski, F. Feingold, R. Blair, Y. O. Halchenko, E. Miller, N. Hardcastle, J. Wexler, O. Esteban, M. Goncavles, *et al.*, "The openneuro resource for sharing of neuroscience data," *Elife*, vol. 10, p. e71774, 2021.

[4] K. J. Gorgolewski, T. Auer, V. D. Calhoun, R. C. Craddock, S. Das, E. P. Duff, G. Flandin, S. S. Ghosh, T. Glatard, Y. O. Halchenko, *et al.*, "The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[5] O. Esteban, C. J. Markiewicz, R. W. Blair, C. A. Moodie, A. I. Isik, A. Erramuzpe, J. D. Kent, M. Goncalves, E. DuPre, M. Snyder, *et al.*, "fmriprep: a robust preprocessing pipeline for functional mri," *Nature methods*, vol. 16, no. 1, pp. 111–116, 2019.

[6] A. Mensch, J. Mairal, B. Thirion, and G. Varoquaux, "Extracting representations of cognition across neuroimaging studies improves brain decoding," *PLoS computational biology*, vol. 17, no. 5, p. e1008795, 2021.

[7] A. W. Thomas, U. Lindenberger, W. Samek, and K.-R. Müller, "Evaluating deep transfer learning for whole-brain cognitive decoding," *arXiv preprint arXiv:2111.01562*, 2021.

[8] U. Mahmood, M. M. Rahman, A. Fedorov, Z. Fu, and S. Plis, "Transfer learning of fmri dynamics," *arXiv preprint arXiv:1911.06813*, 2019.

[9] Y. Zhang, L. Tetrel, B. Thirion, and P. Bellec, "Functional annotation of human cognitive states using deep graph convolution," *NeuroImage*, vol. 231, p. 117847, 2021.

[10] A. Mensch, J. Mairal, D. Bzdok, B. Thirion, and G. Varoquaux, "Learning neural representations of human cognition across many fmri studies," *Advances in neural information processing systems*, vol. 30, 2017.

[11] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Association for Computational Linguistics, 2019.

[13] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, pp. 1–67, 2020.

[14] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[15] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[16] K. Dadi, G. Varoquaux, A. Machlouzarides-Shalit, K. J. Gorgolewski, D. Wassermann, B. Thirion, and A. Mensch, "Fine-grain atlases of functional modes for fmri analysis," *NeuroImage*, vol. 221, p. 117126, 2020.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] D. C. Van Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, W.-M. H. Consortium, *et al.*, "The wu-minn human connectome project: an overview," *Neuroimage*, vol. 80, pp. 62–79, 2013.

[21] E. M. Gordon, T. O. Laumann, A. W. Gilmore, D. J. Newbold, D. J. Greene, J. J. Berg, M. Ortega, C. Hoyt-Drazen, C. Gratton, H. Sun, *et al.*, "Precision functional mapping of individual human brains," *Neuron*, vol. 95, no. 4, pp. 791–807, 2017.

[22] A. L. Pinho, A. Amadon, T. Ruest, M. Fabre, E. Dohmatob, I. Denghien, C. Ginisty, S. Becuwe-Desmidt, S. Roger, L. Laurier, *et al.*, "Individual brain charting, a high-resolution fmri dataset for cognitive mapping," *Scientific data*, vol. 5, no. 1, pp. 1–15, 2018.

[23] L. Snoek, M. M. van der Miesen, T. Beemsterboer, A. van der Leij, A. Eigenhuis, and H. Steven Scholte, "The amsterdam open mri collection, a set of multimodal mri datasets for individual difference analyses," *Scientific data*, vol. 8, no. 1, pp. 1–23, 2021.

[24] N. Chang, J. A. Pyles, A. Marcus, A. Gupta, M. J. Tarr, and E. M. Aminoff, "Bold5000, a public fmri dataset while viewing 5000 visual images," *Scientific data*, vol. 6, no. 1, pp. 1–18, 2019.

[25] S. A. Nastase, Y.-F. Liu, H. Hillman, A. Zadbood, L. Hasenfratz, N. Keshavarzian, J. Chen, C. J. Honey, Y. Yeshurun, M. Regev, *et al.*, "The "narratives" fmri dataset for evaluating models of naturalistic language comprehension," *Scientific data*, vol. 8, no. 1, pp. 1–22, 2021.

[26] M. King, C. R. Hernandez-Castillo, R. A. Poldrack, R. B. Ivry, and J. Diedrichsen, "Functional boundaries in the human cerebellum revealed by a multi-domain task battery," *Nature neuroscience*, vol. 22, no. 8, pp. 1371–1378, 2019.

[27] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962*, 2019.

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[29] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

[30] M.-A. Schulz, B. Yeo, J. T. Vogelstein, J. Mourao-Miranada, J. N. Kather, K. Kording, B. Richards, and D. Bzdok, "Different scaling of linear models and deep learning in ukbiobank brain images versus machine-learning datasets," *Nature communications*, vol. 11, no. 1, pp. 1–15, 2020.

[31] Y. Zhang, N. et Farrugia, and P. Bellec, "Deep learning models of cognitive processes constrained by human brain connectomes," *bioRxiv*, 2021.

[32] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, *Explainable AI: interpreting, explaining and visualizing deep learning*, vol. 11700. Springer Nature, 2019.

[33] A. W. Thomas, C. Ré, and R. A. Poldrack, "Challenges for cognitive decoding using deep learning methods," *arXiv preprint arXiv:2108.06896*, 2021.

[34] R. A. Poldrack, E. Congdon, W. Triplett, K. Gorgolewski, K. Karlsgodt, J. Mumford, F. Sabb, N. Freimer, E. London, T. Cannon, *et al.*, "A phenome-wide examination of neural and cognitive function," *Scientific data*, vol. 3, no. 1, pp. 1–12, 2016.

[35] B. Fischl, "Freesurfer," *Neuroimage*, vol. 62, no. 2, pp. 774–781, 2012.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] see section 3.3 and 3.5
   (b) Did you describe the limitations of your work? [Yes] see section 4
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] see section 4
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] see Appendix A.4
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see sections 2.3, 3.2, 3.5, and Appendix A.9 and A.8
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] We replicated our downstream adaptation analysis for the MDTB dataset once with a different set of random seeds and found almost identical results (see Appendix A.10)
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see section 2.3
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] see section 3.1 and Appendix A.1
   (b) Did you mention the license of the assets? [Yes] see Appendix A.1
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] we make our pre-trained models and training data publicly available; see section A.4
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] see section 4 and Appendix A.1
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] see section 4 and Appendix A.1
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]