Aboutness in Artificial Neural Networks Trained on MNIST

Francesco Lässig

Abstract

Many scientific theories and philosophical frameworks of consciousness assume a functional mapping between brain states and conscious experience. To approach the question of what such a mapping could look like, we take inspiration from structuralist views on consciousness postulating that conscious experience is determined relationally. To test whether it is possible to extract information about what a neural network is representing purely from relational information of its current state (activity and connectivity), we propose the following decoding task: Provided an unseen, newly trained MNIST-classifying multi-layer perceptron with randomly permuted output neurons, can we predict which MNIST digit a given output neuron represents purely based on relational information about the networks connectivity? We show that this is possible using a self-attention decoding architecture and propose that this approach holds potential for being a decoder of conscious contents, or of 'aboutness' in neural networks more generally.

1 Introduction

If we assume narrow representationalism, i.e. the idea that conscious states only supervene on brain states that represent the contents of experience [Rey98], then there has to exist a unique functional mapping from neural activity (or brain states more generally) to conscious content. In other words, the same neuronal activity cannot account for 2 conflicting contents of consciousness. This is not a trivial statement, because it suggests the presence of an overarching principle that determines this mapping. This principle has to answer the question of why my current conscious experience is about some things to the exclusion of others, purely based on my neural activity in the current moment (but allowing for a minimum time window for conscious experience to emerge).

Without necessarily invoking consciousness, the same question of 'aboutness' can be asked for artificial neural network representations. Can we know what a given network representation is about purely by looking at the network state? We can frame this as a decoding problem: Can we construct/train a decoder that predicts stimulus-related information (e.g. stimulus class) purely from network activity and structure? A trivial way to come up with a decoding scheme for intentional content would be to fit a decoding model to map neural network states to stimulus information based on many stimulus information - network state pairs of the same network. While this will work (for example the decoder can trivially learn the output layer of the underlying network to map last hidden layer activity to classes), it cannot be considered a general principle of aboutness. If we retrain the same network with a different random seed, the previously trained decoder is likely going to assign the same neuron activity to a different class.

One idea of what a general principle of aboutness could be is inspired by neurophenomenal structuralism [FKL21]: the idea that the contents of a conscious experience are determined relationally, and that the phenomenal relations are reflected in the neural substrate. The activity of a neural network could be about something by virtue of encoding a web of relations that uniquely characterize a certain set of contents of consciousness. Our idea is that a decoding algorithm could pick up on unique webs of relationships (and positions within those webs) and would generalize across different networks. Our hypothesis is that such relational webs can be recognized by their relational structure alone, independent of which neurons actually code for which part of the structure. To ensure that the decoder doesn't make use of trivial, network-specific correlations of network state and stimulus information, the neural network activities and structures in the training and test data of the decoder will be obtained by training underlying networks using different random seeds, leading to different representations.

We propose the following task: For an unseen network trained to classify MNIST images, we want to deduce the class that a given output neuron encodes, with no guarantee about the order in which the output neurons are given. Moreover, we want to decode the class of a given neuron purely based on the connectivity of the output layer to the previous layer. The idea is that the connectivity of the network allows us to identify a relational structure between the output neurons that reflects characteristics of the MNIST distribution. Within this relational structure, we hypothesize that each MNIST class occupies a unique position relative to the other classes.

2 Machine learning setup

To operationalize this idea, we want to turn it into a machine learning problem: Can we train a decoder that predicts the class of an output neuron purely based on connectivity of the output layer to the previous layer? Note that we cannot train the decoder on the same network it should be evaluated on, since that would result in the decoder learning a network-specific mapping of neurons to classes, and not a general principle of aboutness for MNIST. To avoid this, we train many networks using different random seeds on MNIST to generate the training and validation data. Crucially, data from the same network is only contained in either the training or the validation set, but not in both. This way, the only way for the decoder to solve the task is to learn to recognise consistent patterns in connectivity across different networks that are informative about class identity.

3 Dataset

To train a decoder to predict which class a given output neuron represents based on the incoming weights to the output layer, we construct an input-output pair (X, y) in the following way: For a given network that was trained on MNIST, let W denote its output layer weights (Figure 1a). We define X as a matrix consisting of a random permutation of the rows of W (Figure 1b). This means that each row of X corresponds to the input weights of one of the output neurons of the underlying network. Moreover, any row of X could be associated with any of the output neurons of the underlying network, and thus with any of the 10 MNIST classes. Finally, we define y as the class index of whichever output neuron ended up being the first row of X. Thus, what the decoder 'sees' is a set of weight vectors corresponding to output neurons of the network. The position of these vectors within X contains no information, except that the decoder will have to predict the class index of the one that occupies the first row. For each underlying network trained on MNIST we generate 10 data points, one for each value of y, giving us a total of 10000 data points.

4 Preprocessing

While the dataset we described above should contain all the information the decoder needs to predict the class of an output neuron by identifying its relations to other output neurons (if this relational structure does indeed exist), in practice we found that the decoder does not naturally learn to identify these relations (at least not within the limited training time we used to fit the decoder). To point the decoder into the right direction, we applied the following preprocessing step to X:

$$X' = XX^T \oslash \|X\|_{row} \|X\|_{col}^T$$

$$(X')_{i,j} = \frac{(X_{i,:})^T (X_{:,j})}{\|X_{i,:}\| \|X_{:,j}\|}$$

$\oslash$ denotes the element-wise Hadamard division between two matrices, $\|.\|_{row}^T$ and $\|.\|_{col}^T$ denote the operation that takes the row and column-wise L2-norm of a matrix, respectively, resulting in a vector of scalar norms. Like X, the rows of X' all correspond to one of the output neurons and the first row corresponds to the output neuron whose class index should be predicted by the decoder (Figure 1c). However, instead of representing the incoming weights of an output neuron, a row now represents the angles between that neuron's incoming weights with all other output neuron's incoming weights. In other words, the value $(X')_{i,j}$ represents the angle between the incoming weights of output neuron i and output neuron j. Note that i and j correspond to the indices within X, which was created from a random permutation of W^L. In other words, i and j are not informative of the class indices. However,

X' now encodes the output neurons in terms of their input weights in a much more explicitly relational fashion. In addition to facilitating better decoding accuracy, this has the advantage that the decoder, if successful, identifies classes of output neurons exclusively based on relational information.

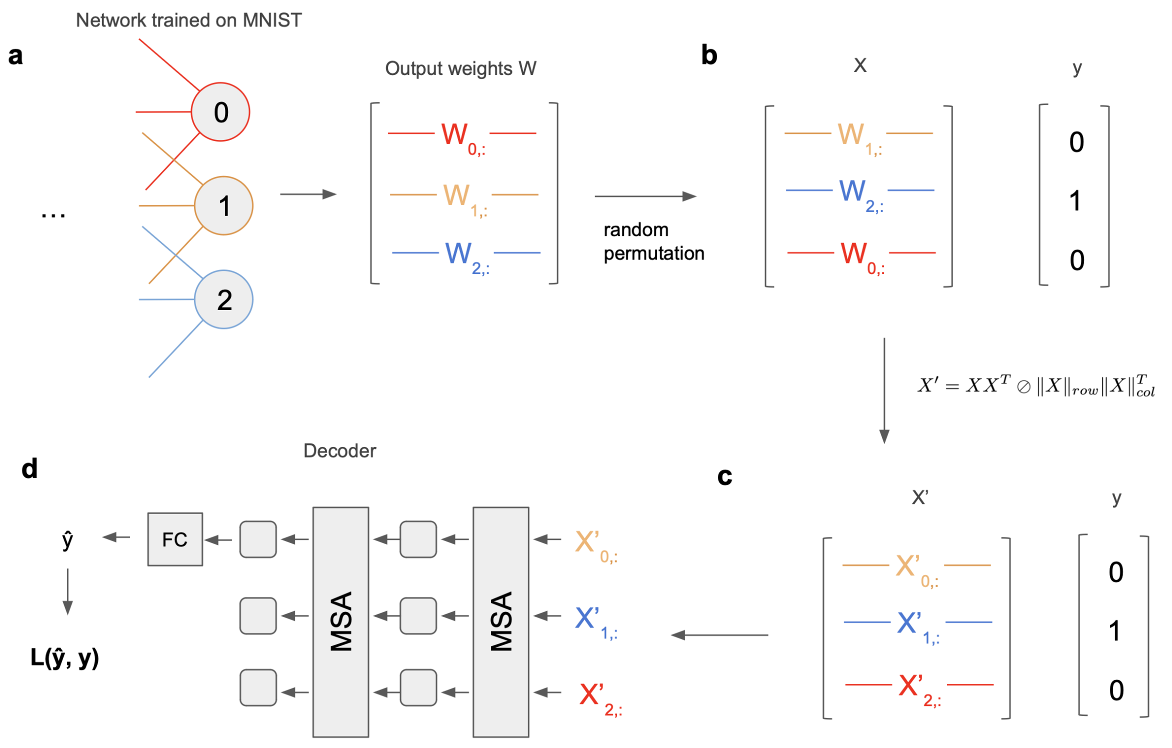

Figure 1: Data processing pipeline from underlying MNIST-trained network to decoder. To simplify the diagrams, we are considering a hypothetical network with only 3 output units. **a:** As a basis for our decoding task we consider the output layer of a fully-connected feedforward network trained to classify MNIST using backpropagation. The connectivity matrix contains the incoming weights for each output neuron in its rows. **b:** To create a data point for the decoder, we permute the rows of the output layer connectivity matrix such that the class identity of an output neuron cannot be determined based on its position in the matrix. The input weights of the output neurons whose class identity should be predicted is in the first row. Hence, in this example, the second element of the target output y is equal to 1 because the original index of the output neuron in the first row of X is equal to 1. **c:** To facilitate extraction of relational information between output neurons, we generate matrix X' which in row i contains the angles between the incoming weights of the i'th output neuron with the weights of all other output neurons. Note that i here corresponds to the new indices after permutation, which means that i is not informative about class identity. **d:** The rows of X' are treated as tokens and fed into a multi-head self-attention (MSA) based decoder network. We pass the data through two MSA layers, after which only the representation of the first token (corresponding to the first row of X', which in turn corresponds to the output neuron whose class we want to identify) is fed into a fully-connected linear layer (FC) which maps to a 10-dimensional space (corresponding to the 10 MNIST classes). Finally, during training, the cross-entropy loss (L) is computed between the prediction $\hat{y}$ and the target value y.

5 Decoder architecture

Because the order of the rows of X beyond the first row (which always corresponds to the output neuron whose class should be predicted) contains no useful information to solve the task, we want our decoder to be invariant to permutations of the rows of X'. We achieve this using a Transformer-like architecture with self-attention layers [VSP+17], as seen in Figure 1d. We treat the rows of X as tokens, pass the data through two multi-head self-attention layers and finally read out the result from the first token's learned representation using a linear layer that produces a 10-dimensional output.

During training, we compute the cross entropy loss between this output and the label y. To compute the validation accuracy, we simply take the output of the decoder with the highest value as our class prediction for a given data point.

6 Results

To evaluate whether the output layers of the underlying MNIST networks encode relational information that allows us to identify the class of output neurons, we train our self-attention based decoder for 100 epochs on 8000 datapoints and validate its accuracy on the remaining 2000. We train the decoder on three different datasets, generated by training fully-connected networks on MNIST in three different training paradigms: no training, normal backpropagation, and backpropagation with dropout. The 'no training' paradigm serves as a control. Since the underlying network connectivity is random, there should be no relevant relational structure in the output weights, and hence the decoder accuracy should be equivalent to random guessing (i.e. 0.1). The results are shown in Figure 2.

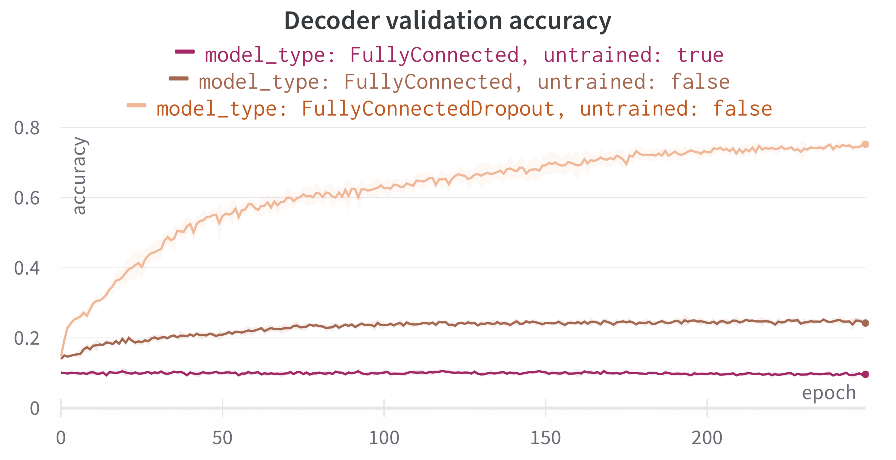

Figure 2: Progression of the validation accuracy during 100 epochs of training the decoder to identify output neuron classes based on an unordered set of weight vectors of output neurons. The error margins reflect the standard deviation across 5 random seeds. We used three different training paradigms to generate the underlying MNIST-trained networks used to generate the data for the decoder: no training, normal backpropagation (FullyConnected), and backpropagation with dropout (FullyConnectedDropout). Note that the 'untrained' flag in the legend refers to the underlying networks used to generate the training data, not the decoder.

We can see that the accuracy of predicting output neuron classes based on their connectivity is above chance level (except for the control dataset of untrained networks, which yields chance-level accuracy as expected). Due to the way we designed our dataset, we can be fairly certain that the decoder achieves this purely based on relational information between output neurons. While training the decoder on the standard MNIST-trained networks (no dropout) yields some correct predictions resulting in a validation accuracy of roughly 25% at the end of training, the final accuracy jumps to about 75% as we switch to the dataset that was produced with dropout. Intuitively we are not surprised that dropout yields higher decoding accuracy, as encourages neurons to rely on population activity rather than single-neuron pathways [BS13]. If output neurons rely on population activity of the last hidden layer, output neurons of similar MNIST digits should also have similar input weights, as they should share more features than output neurons representing dissimilar MNIST digits.

7 Discussion

By achieving a decoding accuracy significantly higher than random guessing, we showed that networks trained on MNIST encode class identity of output neurons relationally in their output weights. We also showed that the extent to which this happens is dependent on the training paradigm used for the underlying networks. It is very likely that other training paradigms would yield even higher decoding accuracies, as we did not perform any optimization of hyperparameters. It is also likely that generative networks could yield much higher accuracies than discriminative networks.

Our experimental setup could easily be extended to decode other representational content of neural networks. For instance, staying with MNIST, given randomly permuted input neurons, one could try to decode their position in pixel space based on their connectivity to the next layer. Or, for natural image tasks, given randomly permuted color channels, one could try to decode the color from an input neuron based on their connectivity to the next layer. Moreover, the same decoding scheme might be applicable to neural data: X' could be chosen as the correlation matrix of recorded neural activity.

While actual connectivity between neurons might not be the deciding factor when it comes to determining conscious content in biological brains, the same idea can easily be extended to other relations, such as activity correlations. In other words, instead of providing the decoder with a connectivity matrix, we could provide it with a matrix of correlations of neuronal activities. Pennartz proposed correlations between neurons to dictate conscious experience by spanning a relational network in 2009 [Pen09].

In the ideal case, self-attention based decoders could generalize to decode representational content across different domains of inputs and across different network architectures, thus presenting a potential avenue for not just a 'consciousness meter', but a 'conscious content decoder'.

References

[BS13] Pierre Baldi and Peter J Sadowski. Understanding dropout. *Advances in neural information processing systems*, 26, 2013.

[FKL21] Sascha Benjamin Fink, Lukas Kob, and Holger Lyre. A structural constraint on neural correlates of consciousness. *Philosophy and the Mind Sciences*, 2, 2021.

[Pen09] Cyriel MA Pennartz. Identification and integration of sensory modalities: neural basis and relation to consciousness. *Consciousness and cognition*, 18(3):718–739, 2009.

[Rey98] Georges Rey. A narrow representationalist account of qualitative experience. *Philosophical perspectives*, 12:435–457, 1998.

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Appendices

A Hyperparameters

In the following we list all hyperparameters that were chosen for the underlying networks to generate the dataset (Table 1), and for the self-attention based decoder (Table 2). Note that none of these hyperparameters were optimized using gridsearch or similar schemes, most of them were chosen quite arbitrarily, since this is only supposed to be a proof of concept.

Name	**Value**
learning rate	0.001
batch size	256
epochs (except for the non-train paradigm)	2
hidden dimensionalities	50, 50
droput rate (only for the dropout paradigm)	0.2

Table 1: Hyperparameters for underlying, MNIST-trained networks used to generate the training and validation data for the decoder. Note that the number of epochs in the 'untrained' paradigm was set to 0, and the dropout rate only applies to the 'dropout' paradigm.

Name	**Value**
learning rate	0.001
batch size	64
epochs (except for the non-train paradigm)	100
hidden dimensionality	64
number of attention heads per MSA layer	4
number of MSA layers	2

Table 2: Hyperparameters for decoder. MSA is short for multi-head self-attention.