PUTTING MORE *NEURAL* IN ARTIFICIAL NEURAL NETWORKS

by

CALLIE P. FEDERER

B.S., Truman State University, 2015

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Computational Bioscience Program

2019

This thesis for the Doctor of Philosophy degree by

Callie P. Federer

has been approved for the

Computational Bioscience Program

by

James Costello, Chair

Joel Zylberberg, Advisor

Larry Hunter

Alon Poleg-Polsky

Abigail Person

Gidon Felsen

Zack Kilpatrick

Date 13 Dec 2019

Federer, Callie P. (Ph.D., Computational Bioscience)

Putting More *Neural* in Artificial Neural Networks

Thesis directed by Assistant Professor Joel Zylberberg

ABSTRACT

In computational neuroscience, we are at the intersection of neuroscience and artificial intelligence (AI). In this thesis, we work towards understanding how can we use AI algorithms as a model of the brain, and how can the brain influence how we design AI algorithms? Throughout this work we use artificial neural networks (ANNs), AI algorithms that were directly influenced by the brain, to solve two problems – working memory and computer vision.

We first consider working memory, which requires information about external stimuli to be stored and represented in the brain for tens of seconds even after the stimuli goes away. Prior work in this field relies on learning rules or network organization that is not biologically plausible. To identify mechanisms through which biological networks can learn memory function, we derived biologically plausible plasticity rules that enable information storing. We then demonstrate these networks' robustness and ability to continue to learn.

We next consider the field of computer vision, where we are trying to get machines to interpret visual information as people do. Specifically, we focus on object recognition, getting machines to identify the objects within an image. State-of-the-art algorithms in object recognition still suffer from interpreting images in a way that is not human-like, leading to unexpected and potentially catastrophic errors. We outline a new way to train object recognition algorithms using the brain as a teacher for training. We also propose a new metric for evaluating the success of our object recognition algorithms by evaluating the human-likeness of the errors.

The form and content of this abstract are approved. I recommend its publication.

Approved: Joel Zylberberg

ACKNOWLEDGEMENTS

This dissertation would not be possible without a significant amount of support from colleagues, friends and family. First I want to acknowledge the amazing mentorship of my advisor, Joel Zylberberg. During my rotation with him, he suggested I could figure out how to solve the problem better than he could. He was of course mistaken, but he gave me the confidence and the freedom to try out my own ideas for the first time. He is extremely supportive and kind despite being alarmingly intelligent. He created a wonderful, collaborative, and fun environment where we felt comfortable sharing our ideas about research or axe-throwing.

Thank you to my fellow graduate school labmates, Shelly Jones ('lab wife') and Elijah Christensen ('lab friend'). For every minute Shelly distracted me away from getting work done, she contributed two minutes towards me surviving this program. Despite the many times Elijah googled my problems for me, he never once suggested I should google it myself.

I am extremely grateful to my thesis committee members: James Costello, Larry Hunter, Alon Poleg-Polsky, Abigail Person, Gidon Felson and Zack Kilpatrick. They have all provided insight and advice for my work throught my graduate career, which I very often needed. And a special thanks to the Person lab for taking me in.

Thank you to our collaborators, Alona Fyshe and Haoyan 'Ed' Xu, who made me actually look forward to skype meetings, always let me share my ideas, and often had even better ideas to contribute and keep me motivated.

Thank you to the Computational Bioscience Program who created a fun community to excel in and for whatever reason let me in this program in the first place.

Thank you to my Truman State University faculty members, especially Jon Beck who guided me towards this graduate program and gave me continuous support and encouragement (and letters of recommendation). Thank you to my boyfriend, Ryan Meinkoth, who moved in with me right before my preliminary exams and still seems to like living with me today. Thank you for letting me cry when you bought me the wrong type of sushi on day 2 of my exams. Thank you to our puppy daughter, Macy, for being you.

Thank you to my family for the continuous support, love, guidance, and humor during this program. I owe my success to my parents who have always been there for me in any and every way possible. Thank you to my brothers for being proud of me.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

CHAPTER I
INTRODUCTION

I.1 *Neural* in Artificial Neural Networks

Before we put some more neural in artificial neural networks (ANNs), let's discuss how much *neural* artificial neural networks have. In other words, how brain-like are ANNs? This ongoing debate helps us consider **1** how much can we use ANNs as a model of the brain? and **2** how much can the brain influence how we design ANNs?

I.1.1 Neuro-inspired Artificial Intelligence

Although artificial intelligence (AI) and neuroscience are often considered separate fields today, AI originated out of neuroscience (Hassabis *et al.* (2017)). The first step towards the artificial neural network was the artificial neuron described in 1943 (McCulloch and Pitts (1943)). The idea being to try and mimic the functionality of a biological neuron. Donald Hebb introduced a theory on the way synapses in neural networks change, which we now call Hebbian learning. The idea behind this theory being that when an input neuron fires, if it frequently leads to the firing of an output neuron, the synapse is strengthened, also referred to as 'Neurons that fire together, write together' (Hebb (1949)). The Perceptron was created with inspiration from the ideas behind the artificial neuron (McCulloch and Pitts (1943)) and how synapses between neurons can change (Hebb (1949)). The Perceptron is a single neuron-like unit that takes in some inputs $(x1, x2, x3)$ and produces an output based on a predetermined function (Fig. I.1 A.). Suppose for example we wanted to learn the differences between apples and oranges. Inputs that would help us learn these difference could be color $(x1)$, sweetness $(x2)$, and shape $(x3)$. The Perceptron then learns how important each attribute by changing the weighting of each input, $(w1, w2, w3)$. The output, y, gives us a value that we can use to interpret if we are more likely to have an apple or an orange. The weighted function gives us a decision boundary of which areas are more apple-like and which are more orange-like. In the simplest case, y can be a linear sum over the weighted inputs, $y = x1 * w1 + x2 * w2 + x3 * w3$. The New York Times called The Perceptron "the embryo of an electronic computer that the Navy expects will be able to walk, talk, see, reproduce itself and be conscious of its existence" (NYT (1958)). The Perceptron was originally created to try to copy how neurons learn in the brain so that we could understand the capability of higher organisms for thinking and learning. It began as a model for how the human brain processed visual data and learned to recognize objects but was picked up by other researchers to study other areas in human cognition.

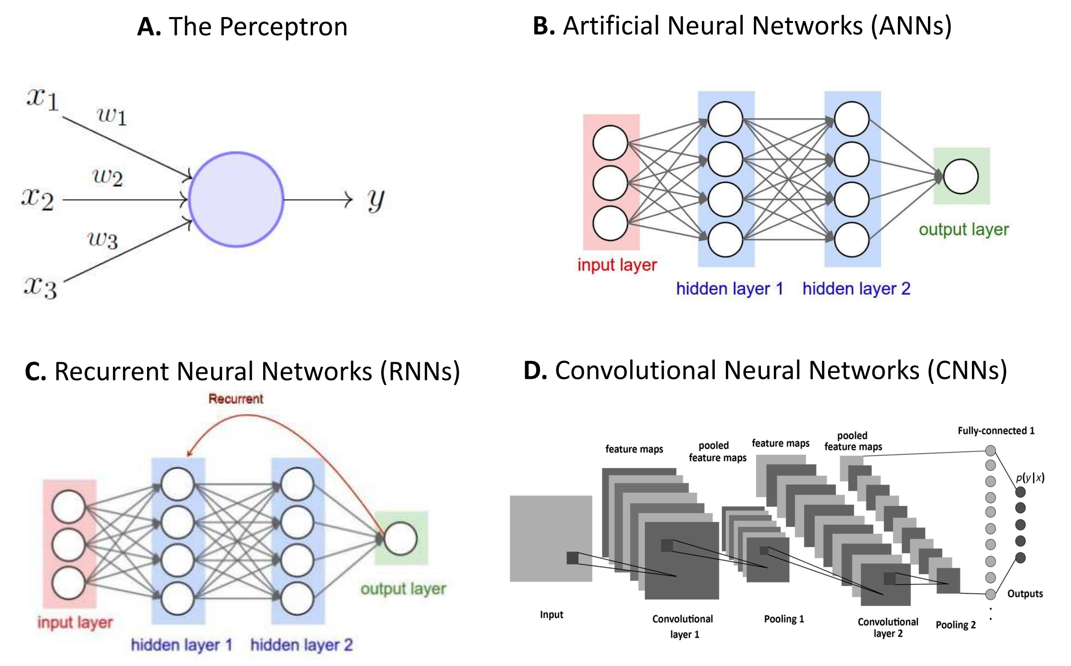


Figure I.1: Neuro-inspired algorithms in artificial intelligence. A. The Perceptron. The percepton incorporates some inputs, $x1, x2, x3$, weighted by $w1, w2, w3$ (which are learned). These are combined based on a function (in the simplest case - linear) so y is a function of these weighted inputs ($y = f(x1 * w1, x2 * w2, x3 * w3)$. B. The artificial neural network (ANN), adapted from The Perceptron used for processing information from one layer to the next. C. On schematic of a recurrent neural network (RNN), which incorporates weights within the layers (noted by the recurrent loop) via a cyclic topology to store memory. D. The convolutional neural network (CNN), used for processing visual information by layers of convolutions or feature maps. These images were adapted from Minsky-Papert 1969, Luke Dormel, Emerging Tech 2019 and Alex Yu, Towards Data Science 2018, respectively.

If we wanted to incorporate a more complex algorithm, such as being able to categorize all possible fruits, we would need more of these neuron-like units. This leads us to the artificial neural network (ANN). ANNs are computing systems that are layers of neuron-like units, like The Perceptron (Fig. I.1 B.). As we've seen now from many applications of ANNs, researchers realized it could be utilized for solving real-world problems, regardless of how 'brain-like' it was.

Artificial neural networks are typically trained by an algorithm called 'backpropagation of errors' or 'backprop'. Neural networks are initialized with random weights, meaning the mapping from input to output initially is a random, and likely poor, estimate. To train networks, we input some data (x) to the network that gets fed to subsequent layers of the network weighted by the randomly initialized weights, w. We then get an output, or prediction, y. We calculate the error based on how different the network output, y is from the correct answer we'll denote as $\hat{y}$. The error is then backpropagated and each weight is adjusted based on its contribution to the error. While backpropagation is not plausible to occur in actual brains, it can give us a starting point on how brains might be learning (more on this discussed in the next section). State-of-the-art deep learning algorithms have largely stemmed from artificial neural networks, created in order to mimic how the brain learns (Rosenblatt (1958)). In this work, we utilize two variants of ANNs.

Recurrent neural networks (RNNs), used in ch. II, are networks of neuron-like units that are connected and form a directed graph and can exhibit temporal dynamic behavior (Rumelhart *et al.* (1986a)) (Fig. I.1 C.). As in ANNs, the weights describing the connections between the neurons are repeatedly adjusted so that the network output more closely matches the desired output. RNNs differ from ANNs in that they do not only feed information forward to the next layer of inputs but maintain an internal state (memory) of previous inputs. This is noted by the red line 'recurrent' leading from the output layer to hidden layer 1 (Fig. I.1 C.). This internal memory is a function of connectivity loops within the layers, meaning within each layer, neurons can 'talk' (or feed information) to other neurons in the same layer allowing memory of past inputs. These temporal dynamics are also important in neuron cells.

Convolutional neural networks (CNNs), used in ch. III, are networks of neuron-like units that are primarily used for processing visual information (Fig. I.1 D.). CNNs were inspired by the architecture of the mammalian visual system and process and respond to a restricted region of the visual field or receptive field (Hubel and Wiesel (1968)). The input to CNNs are typically images (or videos) in matrix form (see input Fig. I.1 D.). The convolutional layers are learned throughout training to pull out important aspects of the input images. Earlier layers of the network typically learn simple features, such as where horizontal and vertical lines are. The pooling layers further reduce the convolutional layers and give a simple map of where the features of interest are. Later convolutional layers pull out more detailed parts of the images, for example some neurons in the later convolutional neural layer may respond only to peoples' faces. In CNNs trained for object recognition as done in III, the output layer gives a probability function of which class the input image most likely belongs to (i.e. 'cat' or 'dog').

I.1.2 Theoretical Neuroscience

Studying methods in how machines can learn helps offer insights that can be applied in neuroscience. AI provides ideas on how the brain can learn via backpropagation, the method used for training ANNs. The idea being that weights within multiple layers of a hierarchical network need to be optimized toward some objective function (Rumelhart *et al.* (1986a)). There are still many aspects of backpropagation that are not biologically plausible. The main objections to backpropagation being able to occur in the brain include: (1) the need for symmetric weights so that the feedback connections carrying the gradient have the same weights as the corresponding feedforward connections and (2) the need for a distinct form of error feedback that does not influence neural activity which currently does not align with any known biological feedback mechanisms (Bartunov *et al.* (2018); Lillicrap *et al.* (2016)). Important work is to understand how aspects of backpropagation could be relaxed or replaced with biologically plausible mechanisms, which we highlight in chapter II.

Further, the neural networks as discussed in this thesis have be used as models for the brain. Recurrent neural networks, which incorporate the 'loops' that occur within neural networks, can be used to model cognitive functions such as decision making processes and working memory (Murray *et al.* (2017b)). Convolutional neural networks have yielded insights into the mammalian vision representations (Yamins and Dicarlo (2016); Kriegeskorte (2014)). Machine learning techniques are also utilized in analyzing neuroscience data, for example multivariate analysis of fMRI and MEG data for understanding and classifying disease (Kriegeskorte and Kievit (2013)). The work here is focused on two problems in neuroscience and AI and how we can make progress in both fields by incorporating more biological constraints in our machine learning tools.

I.2 Working Memory

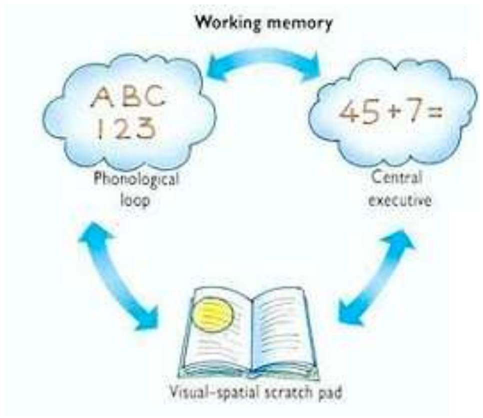


Figure I.2: Baddeley and Hitch's model of working memory, simplified with examples. Adapted from ES Fein. HD Visual Working Memory Separates the Young and Old, 2014.

Working memory is a key cognitive function which is responsible for temporarily holding information available for processing. It is often used as a synonym for short-term memory but encompasses both the storing and the manipulating of information, where as short-term memory suggests only storing. Baddeley and Hitch (1974) introduced the multicomponent model of working memory which highlights the importance of humans being able to store and manipulate multiple items at once, unlike previous models (Atkinson and Shiffrin (1968)). In this multicomponent model, the central executive is responsible for directing attention to relevant information to remember and ignoring less important details (Fig

I.2). The phonological loop, visuo-spatial sketchpad and episodic buffer are controlled by the executive function and store the information. The phonological loop stores sounds such as speech and music. The visuo-spatial scratchpad stores visual and spatial information. The episodic buffer, which was added later, stores representations that integrate phonological and visuo-spatial information and is the link between working memory and long-term memory (Hatfield (2010)).

I.2.1 Working Memory and Intelligence

Working memory is key for reasoning and decision making (Diamond (2012)), both of which are required for success and are potential indicators of intelligence. Higher capacity of working memory may be correlated with greater intelligence (Cowan (2010); Schewizer and Moosbrugger (2004)). Intelligence, in this case, is measured based on IQ tests (Sternberg and Berg (1986)) and the Advanced Progressive Matrices (Raven (1962)) which measure abstract reasoning and is used as a non-verbal estimate of fluid intelligence, or the capacity to reason and solve novel problems. The average human working memory capacity is said to be 4 (Cowan (2010)), meaning that we should be able to store about 4 different pieces of information for a short time and be able to manipulate each piece. For example, credit card numbers are often grouped in fours to make it easy to read them off. In studies comparing working memory capacity in 7-year-olds versus adults (>18 years old), capacity was found to be higher in adults (Cowan (2010)), suggesting that changes in capacity are correlated with changes in intelligence. The reason we have a limited capacity of information we can store is likely because it is expensive to use more brain resources to store more items if four is sufficient for survival. Alternative work has proposed that working memory might be better conceptualized as a limited resource distriubuted among all items, meaning the quality rather than quantity of working memory representations would determine performance Ma *et al.* (2014). One question then is if having unlimited working memory (in respect to capacity or quality), which is much more feasible in artificial intelligence robots than humans, would lead to super-human intelligence.

I.2.2 Deficits of Working Memory in Disease

Many diseases are associated with a decline in working memory. As working memory is a key cognitive function, decline in working memory is extremely disruptive to day-to-day life. Symptoms associated with working memory deficits include difficulty in dividing attention and manipulating remembered information leading to an inability to keep track of conversations (Alberoni *et al.* (1992)). Many behavioral disorders are associated with deficits in working memory, including attention deficit/hyperactivity disorder (ADHD) and oppositional-defiant disorder (ODD) (Engelhardt *et al.* (2017); Skogan *et al.* (2014)). A wide range

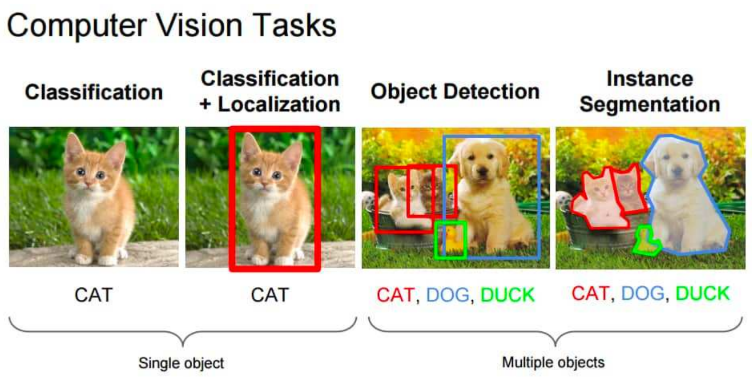


Figure I.3: Adapted from Fei-Fei Li, Andrej Karpathy & Justin Johnson (2016). An overview of computer vision tasks including classification, localization, detection and segmentation.

of other diseases also note deficits in working memory as a symptom, including Alzheimer's disease, Parkinson's disease, and hypertension (Huntley and Howard (2010); Possin *et al.* (2008); Murphy *et al.* (2010)).

In many of these diseases, it is unclear what part of working memory is malfunctioning or how to address these issues. Understanding the mechanisms behind working memory can benefit those with deficits as well has help characterize and better understand diseases. We can use models of working memory to better understand the relationship between disease and decline in working memory.

In chapter II, we derive biologically realistic learning rules that train RNNs to perform working memory tasks which can be used to better understand the mechanisms of working memory.

I.3 Computer Vision

Computer vision is the field of getting computers to be able to interpret images and videos as well as (or better than) people. To interpret images and videos requires identifying the objects being able to abstract out how each object is interacting with each other, which first requires being able to classify the objects and localizing where in the image the objects occur (Fig. I.3). Applications of computer vision include scene reconstruction, event detection, motion tracking, object recognition, 3D pose estimation, motion estimation and image restoration (Morris (2004)). Computer vision is key for the future of self-driving cars, being able to detect objects around the vehicle and predict where objects will be next in order to safely drive. Computer vision will be required for assistance robots that can do things like deliver our Amazon packages or take care of the elderly. We can also use computer vision to evaluate medical images, such as cancer tissues or fMRI results.

Lawrence Roberts, who calls himself the founder of the Internet, is better regarded as the father of computer vision. His dissertation work at MIT discusses extracting 3D information about solid objects from 2D photographs (Roberts (1963); Zdziarski (2018)). Soon after this work, “The Summer Vision Project" took place at MIT in 1966 with the goal of having computers describe what they see (Seymour (1966)). This started the foundation of algorithms used today for computer vision, as well as a better understanding of how difficult this problem would be. After discovering real-world images were much more difficult than expected, work focused on labeling where lines where in order to discern a shape (Huffman (1971); Clowes (1971); Zdziarski (2018)). The next key step towards computer vision algorithms was from David Marr’s book “Vision: a computational investigation into the human representation and processing of visual information" where he proposed that the key to understanding images is by multiple low-level processing algorithms (Zdziarski (2018)). Finally the ‘neocognitron’ was born, the origin of the Convolutional Neural Network (CNN) architecture, introduced by Kunihiko Fukushima in 1980 (Fukushima (1980)). It was inspired by the architeccture of cat and monkey visual cortexes described by Hubel and Wiesel in the 1960s Hubel1968. From this original work, architectures in CNNs led to great advancements in the field of computer vision.

In chapter 3, we show how data from the monkey brain can be used to help train CNNs to better perform on object recognition tasks. In chapter 4, we present a new dataset that can be utilized to better evaluate and even train these computer vision algorithms.

CHAPTER II
A SELF-ORGANIZING SHORT-TERM DYNAMICAL MEMORY NETWORK

II.1 Summary

Working memory requires information about external stimuli to be represented in the brain even after those stimuli go away. This information is encoded in the activities of neurons, and neural activities change over timescales of tens of milliseconds. Information in working memory, however, is retained for tens of *seconds*, suggesting the question of how time-varying neural activities maintain stable representations. Prior work demonstrates the connectivity requirements that allow information to be retained for periods much longer than the timescale of individual neuronal activities. The prior work, however, requires precisely constructed synaptic connectivity matrices, without explaining how this would arise in a biological neural network. To identify mechanisms through which biological networks can self-organize to learn memory function, we derived biologically plausible synaptic plasticity rules that dynamically modify the connectivity matrix to enable information storing [1]. Networks implementing this plasticity rule can successfully learn to form memory representations even if only 10% of the synapses are plastic, are robust to synaptic noise, and can represent information about multiple stimuli.

II.2 Introduction

Working memory is the ability to store and manipulate information on short-term time scales and allows us to make decisions (Diamond (2012)). These skills are required for success in school, in social situations and in the work place. Symptoms associated with working memory deficits include difficulty in dividing attention and manipulating remembered information leading to an inability to keep track of conversations (Alberoni *et al.* (1992)). Many behavior disorders including ADHD are in part characterized by a deficit in working memory (Engelhardt *et al.* (2017)). Deficits in working memory are also common in patients with Alzheimer's disease (Huntley and Howard (2010)). Understanding the mechanisms behind working memory can benefit those with deficits as well as help characterize and better understand associated neurological disease.

Working memory relies on us retaining representations of external stimuli even after they go away. Stimulus-specific elevated firing rates have been observed in the prefrontal cortex during the delay period of working memory tasks, and are the main neural correlates of working memory (Funahashi *et al.* (1993); Fuster and Alexander (1971)). Perturbations to the delay period neural activities cause changes in the animal's subsequent report of the remembered stimulus representation (Li *et al.* (2016); Wimmer *et al.*

[1]Portions of this chapter have been published with permission from *Neural Networks* (Federer and Zylberberg (2018))

(2014)). These elevated delay-period firing rates are not static but have time-varying dynamics with activities changing over timescales of tens of *milliseconds* (Brody *et al.* (2003a); Romo *et al.* (1999)), yet information can be retained for tens of *seconds* (Fig. II.1). This suggests the question how time-varying neural activities keep representing the same information.

Prior work from Druckmann and Chklovskii (2012) shows that, if the neural dynamics are in the "null space" of the representation - so that changes to neural activity do not affect the downstream read-out of stimulus information - then information can be retained for periods much longer than the time-scale of individual neuronal activities (called the FEVER model; Fig. II.2 B). That model has a severe fine-tuning problem, discussed below. We identified a synaptic plasticity mechanism that overcomes this fine-tuning problem, enabling neural networks to learn to form stable representations.

While the dynamics of neurons in the FEVER model match that which is observed in the monkey prefrontal cortex during a working memory task (Murray *et al.* (2017a)), the model itself requires that the network connectivity matrix have one or more eigenvalues very near to unity. This will almost surely not happen in randomly-connected networks: fine-tuned connectivity is needed (Zylberberg and Strowbridge (2017), *Gershgorin Circle Theorem*, *Girko's Circular Law*). Druckmann and Chklovskii (2012) suggest a mechanism of Hebbian learning by which this connectivity can be learned. That mechanism requires the read-out weights to form a 'tight frame', meaning the transpose of the weights is identical to the read-out weights, which poses another fine-tuning problem and will not necessarily be true in biological circuits. Thus, the prior work leaves it unknown how synaptic plasticity can form and/or maintain functional working memory networks. Here, we identify biologically plausible synaptic plasticity rules that can solve this fine-tuning problem without making strong assumptions like 'tight frame' representations.

Related work has introduced learning rules that train recurrent networks to track different dynamical inputs (Bourdoukan *et al.* (2012); Bourdoukan and Deneve (2015); Deneve *et al.* (2017); Brendel *et al.* (2017)), and to maintain short-term memory representations (Vertechi and Machens (2014)), within the neural activity patterns. The majority of those works updated the network synapses based on the difference between the decoded network output, and an external signal (Bourdoukan *et al.* (2012); Bourdoukan and Deneve (2015); Deneve *et al.* (2017); Brendel *et al.* (2017)): this is different from the work presented here, where the errors driving plasticity are the changes in the network representation over time, and are thus purely determined by the network output, with no reference to external signals. One key exception is the work of (Vertechi and Machens (2014)), which used unsupervised learning – analogous to our approach – without reference to external “teacher" signals providing global error signals across the network. That work used networks with two different time-scales of synapses, whereas the networks we present here are simpler, requiring only one synapse type. Our networks are also able to

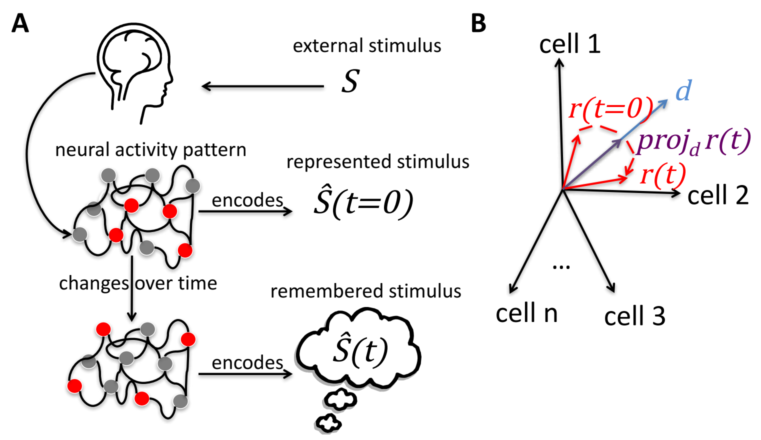


Figure II.1: Stimulus representation in working memory. **(A)** When presented with an external stimulus, s, neural activity patterns initially encode an internal representation of that stimulus, $\hat{s}(t{=}0)$. These neural activity patterns change over timescales of tens of milliseconds, and yet somehow the same information is stored for up to tens of seconds. **(B)** While the firing rates, $r_i(t)$, change over time, information about stimulus, $\hat{s}(t)$, can remain unchanged as long as the projection of the firing rates onto the "read-out" vector, $\vec{d}$, remains constant.

continue to store information without continuous training, whereas Brendel *et al.* (2017) describe the need for continuous training for the recurrent weights in the network. Finally, we demonstrate robustness properties of our networks that are not studied in the prior work: this leads to potentially interesting future work that we describe further in the discussion.

Our plasticity rules dynamically re-tune the connectivity matrix to enable persistent representations of stimulus information. We discuss possible biological mechanisms for networks to implement our plasticity rules. We specifically address parametric working memory, where the requirement is to remember continuous values describing several different variables or dimensions such as spatial locations or sound frequencies. For example, in the oculomotor delayed response task, subjects must remember the location of a target during a delay period (Funahashi *et al.* (1993)). We perform experiments to demonstrate that networks using our plasticity rules are able to store information with added synaptic noise, partial plasticity, and with densely or sparsely connected networks. Our networks are also able to continue to store information even after plasticity has been 'turned off'. We also demonstrate that these plasticity rules work in spiking leaky integrate and fire networks.

II.3 Model

II.3.1 The Rate-Based Network Model

We use a rate-based network like that of the FEVER model (Druckmann and Chklovskii (2012)) but with positive rectifying activation functions (ReLU - rectified linear units) (Carandini (2004)). We use standard linear dynamics in the network model:

$$\tau \frac{da_i}{dt} = -a_i(t) + \sum_j L_{ij} r_j(t), \tag{II.1}$$

where $a_i(t)$ is the internal state (membrane potential) of the ith neuron at time t, $r_j(t)$ is the output (firing rate) of the jth neuron, τ the time constant, and L_{ij} represents the strength of synapse from neuron j to neuron i. Firing rates, $r_j(t)$, are given by a positive rectifying function of the internal states: $r_j(t) = [a_j(t)]_+$. After an external stimulus, s, is presented to the network, the network's representation of the stimulus, $\hat{s}(t)$, is obtained from a weighted combination of firing rates:

$$\hat{s}(t) = \sum_i d_i r_i(t), \tag{II.2}$$

where d_i is the weight of the contribution of neuron i to the stimulus (the "read-out weight"). The initial stimulus value, $\hat{s}(t=0)$, is a weighted combination of a random initialization of the firing rates $\vec{r}(t=0)$. We first consider a single scalar stimulus value, and study the encoding of multiple stimuli in sec. II.4.6

II.3.2 Plasticity Rules

To organize the network, we update the synaptic weights, L_{ij}, so as to minimize changes in the stimulus representation. To do this, we differentiated Equation II.2 with respect to time, to calculate $\frac{d\hat{s}}{dt}$. We used gradient descent with respect to L_{ij} on loss function $(\frac{d\hat{s}}{dt})^2$ to calculate the update rule:

$$\Delta L_{ij} = -\eta \frac{d\hat{s}}{dt} d_i \frac{dr_i}{da_i}(t) r_j(t), \tag{II.3}$$

where η is the learning rate of the network and $\frac{dr_i}{da_i}(t)$ is the slope of the (ReLU) activation function which is one for all positive values of a_i, and 0 otherwise. In section II.4.7, we discuss potential biological mechanisms that allow a network to implement these plasticity rules. We chose the elements of $\vec{d}$ to be positive based on candidate sources of the global error signal discussed in section II.4.7. The source code and scripts for reproducing our experiments are available at https://github.com/cfederer/SOMN.

Our learning rule is similar those for the “slow" weights in recent studies (Bourdoukan and Deneve (2015); Deneve *et al.* (2017)): the learning rules in those prior studies updated synapses based on the product of an error term, and the presynaptic neural activity. In case of linear neurons (e.g., where $\frac{dr_i}{da_i}$ is a constant scalar), and in which we identify $\frac{d\hat{s}}{dt}$ as the error term, our rule reduces to theirs. However, it is important to note that Bourdoukan and Deneve (2015) and Deneve *et al.* (2017) studied supervised learning problems, where the errors are differenced between network outputs and externally-defined target signals. In our case, for contrast, the errors ($\frac{d\hat{s}}{dt}$) are defined purely based on the network outputs, with no reference to external signals.

II.4 Main Results

II.4.1 Stimulus Retention

To evaluate the performance of our working memory networks, we asked how well the networks could store information about a scalar stimulus value. We quantified the remembered value relative to initial, $\frac{\hat{s}(t)}{\hat{s}(t=0)}$, over 3 seconds, where 1.0 indicates perfect memory, and values above or below indicate failure to remember the initial value. We evaluate over 3 seconds because this is the duration of heightened activity observed during working memory tasks (Funahashi *et al.* (1993)).

The networks were all-to-all connected, autapses excluded (so diagonal elements of L_{ij} are all zero) and contain 100 neurons. We chose 100 neurons to make repeat experiments reasonable, however the network works with various network sizes (see sec. II.5.1). Elements of L_{ij} were drawn from Gaussian distribution mean 0 standard deviation $\frac{1}{\sqrt{N}}$ where N = 100 (the number of neurons) (Figs. II.2 A-C). We also considered networks with initially stronger synaptic strengths, where elements of L_{ij} were drawn from Gaussian distribution mean 0 standard deviation $\frac{1.6}{\sqrt{N}}$ (Figs. II.2 D-F) (the network is initially in a chaotic state, as discussed by Toyoizumi and Abbott (2011)). We evaluated how well random networks without plastic synapses store information, by initializing each network with random neural activities, $\vec{a}(t=0)$, random read-out weights, $\vec{d}$ and simulating the dynamical evolution of the representation. The values of $\vec{a}(t=0)$ and $\vec{d}$ were drawn from a Gaussian distribution of mean 0 and variance 1. We then compared these ”constant random synapse” networks to ones with identical random initial conditions, but in which our plasticity rule (Eq. II.3) dynamically updated the synapses. Finally, we compared both of these randomly-initialized networks to ones that had the fine-tuned connectivity specified by the FEVER model.

The stimulus value in the randomly-initialized plastic network initially decreased slightly, but the plasticity rules quickly reorganized the connectivity, and the representation remained constant after the

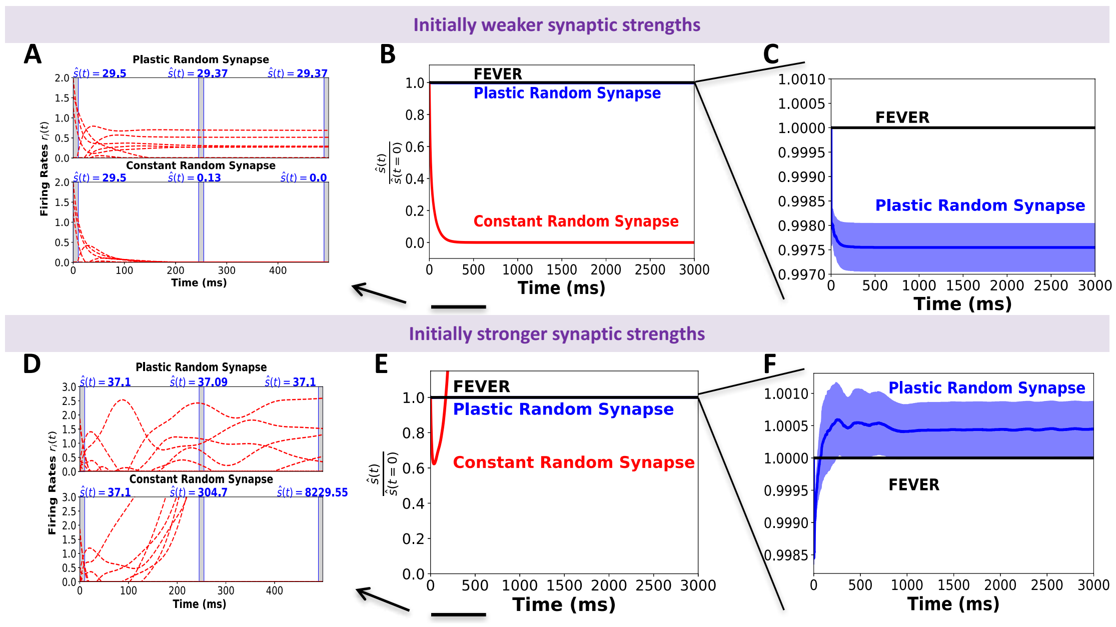


Figure II.2: Stimulus retention in self-organizing memory networks with initially weaker synaptic connections (A-C) where synaptic strengths were drawn from Gaussian distribution mean 0 standard deviation $\frac{1}{\sqrt{N}}$ where N = 100 (the number of neurons) and with initially stronger synaptic connections (D-E) where synaptic strengths were drawn from Gaussian distribution mean 0 standard deviation $\frac{1.6}{\sqrt{N}}$. **(A)** Neural dynamics of the first 500 ms of a simulation of two different networks with the same random initial conditions. One network had constant random synapses (bottom panel), while the other one had plastic synapses that were updated via Eq. II.3 (top panel). Red dashed lines show randomly selected firing rates of 10 of the 100 neurons in the network. The remembered stimulus values at times 0, 245 and 490 ms (indicated by the shaded vertical bars) are shown. In the plastic random network, the remembered stimulus value does not change much **(B)** The average remembered value relative to the initial stimulus value over 10 random initializations for the FEVER model (black), the plastic synapse model (blue) and the constant random synapse model (red) where 1.0 indicates perfect memory. **(C)** A zoomed in look at remembered value relative to initial for the FEVER model and the plastic random synapse model from (B). **(D-F)** Same as A-C, but with stronger initial synaptic strengths. Shaded areas in B, C, and F represent ± standard error of the mean. Shading omitted from lines in panel E for clarity. The "constant random synapse" curve in panel E increases beyond the scale of the plot after around 200-300 ms, as do the neuron activities in panel D (lower).

first ~ 50 ms (Figs. II.2 A, D). In the network with initially weaker connectivity, the firing rates in the plastic random synapse network become constant, whereas in the network with initially stronger connectivity the firing rates in the plastic random synapse network continue to evolve (Figs. II.2 A, D). We considered whether our network evolves to have more static firing rates over time, and simulated the changes in firing rate ($\frac{dr}{dt}$) for neurons in networks with our plasticity rule, that had various initial synaptic strengths (see Additional Results). We found that, while the neuronal activity is more dynamic in networks with stronger synaptic strengths, the network activities nevertheless become more static over time. This may in part be due to the neurons' thresholding nonlinearities: as neurons' membrane potentials drop below threshold, their firing rates become zero, and because the ReLU nonlinearity is flat at this point, changes in membrane potential no longer lead to changes in firing rate.

Within our networks, the synapses change slightly in the initial steps of learning and then stop changing once the network has learned a good connectivity matrix (Fig. II.3). Thus, our plasticity rule enables initially random networks to quickly become effective working memory systems.

To ensure that the success of our plasticity rule at forming an effective memory network was not limited to a fortuitous random initialization, we quantified the fraction of stimulus retained over 10 different networks, each with a different initial connectivity matrix, read-out vector, and initial activity vector $\vec{a}(t=0)$. In the FEVER networks, stimulus retention is perfect across all networks. The models with plastic random synapses perform almost as well as the FEVER models, but require some time to self-organize before the representations remain constant (Figs. II.2 B, C, E, F). In the random constant synapse networks, information is quickly lost (Figs. II.2 B, E).

We demonstrate that the network learns to store information with initially weaker or stronger initial synaptic weights. With the initially weaker weights, the plasticity rule causes the network to learn to generate constant neural activities, after some initial transient (Fig. II.2 A), whereas with initially stronger weights, the network continues to generate complex dynamical activity patterns (Fig. II.2 D). In either case, the neural activity patterns continue to represent the stimulus in a constant fashion, thereby forming effective working memory systems. We first focus on the networks with initially weaker synaptic strengths (e.g., the ones shown in Figs.II.2 A-C, where elements of L_{ij} were drawn from Gaussian distribution mean 0 standard deviation $\frac{1}{\sqrt{N}}$). The corresponding results for networks with initially stronger synaptic strengths (e.g., the ones shown in Figs. II.2 D-F, where elements of L_{ij} were drawn from Gaussian distribution mean 0 standard deviation $\frac{1.6}{\sqrt{N}}$) are shown in section II.5.4.

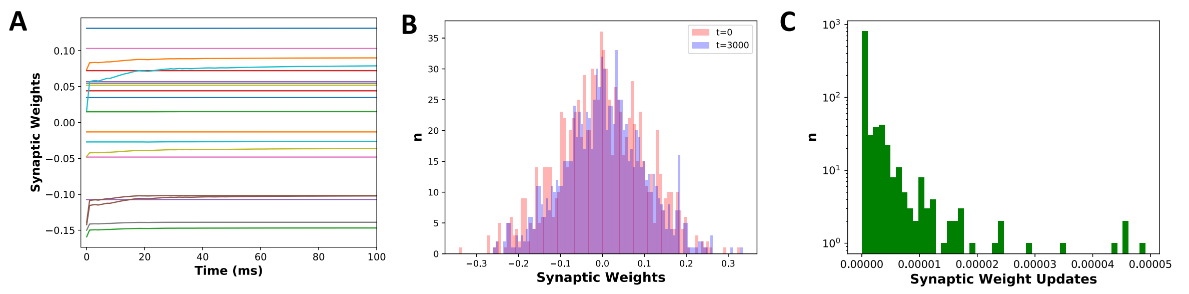


Figure II.3: Synaptic weights in the plastic random synapse networks. **(A)** Strengths of 20 randomly selected synaptic weights over training for the first 100ms of one network initialization. The weights have initial changes but level out at around 50ms once the network has learned to store information. **(B)** A histogram of 1000 randomly sampled synaptic weights from the 10000 in the network before training (t=0, red) and after training (t=3000, blue). **(C)** A histogram of 1000 randomly sampled synaptic weight updates during first step of training for one initialization of a plastic random network presented on a log scale.

II.4.2 Changes in Synaptic Strength

To visualize the effects of our learning rule in the synaptic weights of our network, we randomly selected 20 of the synaptic weights and plotted training for the first 100ms of one network initialization (Fig. II.3 A). The network makes adjustments to the synaptic weights in the initial steps but quickly learns a good connectivity matrix to store information. To demonstrate the magnitude of changes in synaptic weights, we show a histogram of 1000 randomly sampled weights before training (t=0, red) and after training (t=3000, blue) (Fig. II.3 B). The distribution of synaptic weights remains similar. We also plot the histogram of the actual synaptic weight updates (Fig. II.3 C).

II.4.3 Network Robustness

II.4.3.1 *Noisy Synapses*

Working memory must be robust to noise and imprecise components (Brody *et al.* (2003b)). We added Gaussian noise with mean 0 and standard deviation .00001 to the synaptic weights, L_{ij}, in Eq. II.1, to demonstrate that FEVER networks with slightly mistuned synapses quickly forget the stimulus representation, whereas networks with plastic synapses are robust to added synaptic noise (Fig. II.4 A). This emphasizes that fine-tuning is needed in order for the networks to be robust to synaptic noise, and that our plastic networks are capable of overcoming that difficulty. However, a natural question is whether the synaptic *updates* themselves must be fine-tuned, or whether imprecise synaptic updates suffice to maintain the working memory function.

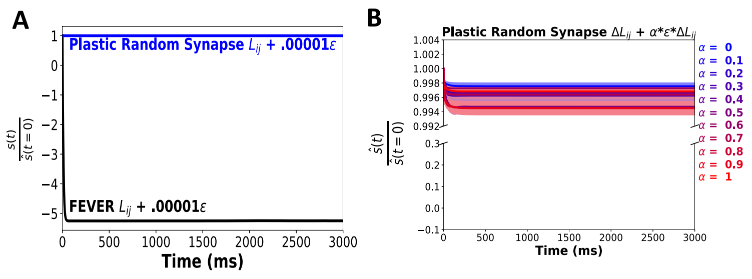


Figure II.4: Network robustness to synaptic noise. **(A)** The average remembered value relative to initial retained for one random initialization of a FEVER network (black) and plastic random network (blue) with independently and identically distributed (i.i.d.) noise added to all of the synaptic weights. The variable ϵ is Gaussian distributed, with mean 0 and standard deviation 1; the noise added to the synapses in panel A has standard deviation of 0.00001. **(B)** The average remembered value relative to initial for 10 random initializations of networks with differing levels of synaptic update noise ($\alpha \leq 1$). Shaded areas in A and B represent $\pm$ standard error of the mean over the 10 different random initializations.

To determine how robust these plastic random networks are to noisy synaptic updates, we simulated networks with various levels of noise added to the synaptic updates in Eq. II.3. We added Gaussian noise with mean 0 and standard deviation α times the update to the synapse, ΔL_{ij}, where α was varied from 0 to 1. (See sec. II.4.2 for synaptic weight update sizes). We quantified the average remembered value relative to initial for networks with various noise levels and found that the noise did not have a noticeable effect on the network performance for $\alpha \leq 1$ (Fig. II.4 B). The remembered value relative to initial is almost equivalent for all values of $\alpha \leq 1$, with minor differences due to random initial conditions. Networks with initially stronger synaptic strengths are also robust to synaptic noise (Fig. II.14). This should not be surprising considering that multiplying error signals by random synaptic weights does not hinder learning, so long as the network is still being pushed down the loss gradient (Lillicrap *et al.* (2016)).

II.4.3.2 *Partial Tuning*

While synapses are plastic, it is not known if *all* synapses change. To determine how well the network performs if only some synapses are updated, we simulated networks in which different fractions of the synapses were updated using Eq. II.3: the other synapses were held constant. We quantified the remembered value relative to initial for these networks (Figs. II.5 A, B). Even with just 10% of the synapses being tuned, the networks learn to store information about the stimulus. Networks with initially stronger synaptic strengths are similarly robust to partial plasticity (Fig. II.15). In hindsight, this makes sense. To store n stimulus values, n constraints must be satisfied by the connectivity matrix (we chose $n = 1$ for Figs. II.5 A, B). Because the connectivity matrix has many more than n elements, many

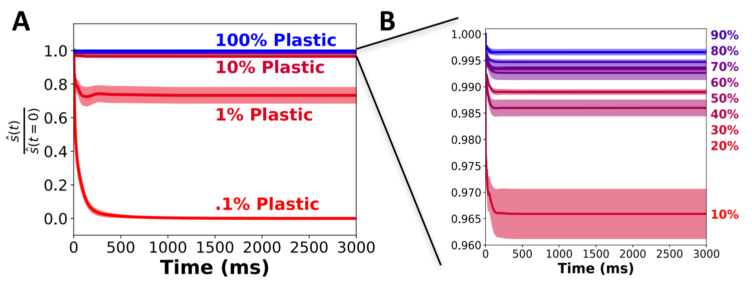


Figure II.5: Network robustness to partial plasticity. **(A)** The average remembered value relative to initial for 10 random initializations. Different lines are for different fractions of plastic synapses. **(B)** A zoomed in look at the average remembered value relative to initial for 10% – 90% plasticity. Shaded areas in A and B represent ± standard error of the mean over the 10 different random initializations.

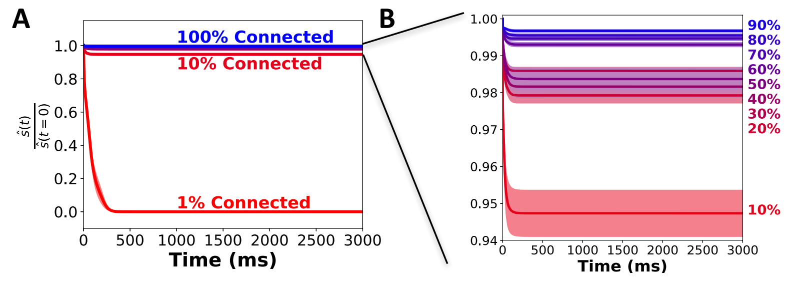


Figure II.6: Network robustness to partial connectivity. **(A)** The average remembered value relative to initial for 10 random initializations. Different lines are for different connection probabilities. **(B)** A zoomed in look at the average remembered value relative to initial for 10% – 90% connection probability. Shaded areas in A and B represent ± standard error of the mean over the 10 different random initializations.

configurations can satisfy the constraint, so it is possible to satisfy the constraint without updating every synapse.

II.4.4 Partial Connectivity

In the previous results, the connectivity was all-to-all (100%). Real neural circuits are not 100% connected. In visual cortex, for example, connection probabilities range from 50% to 80% for adjacent neurons (Hellwig (2000)). To ask if our synaptic update rule could self-organize partially connected networks, we simulated networks with different connection probabilities and with synapses updated using Eq. II.3. In our simulations, each neuron had an equal probability of connected with another neuron in the network. In the brain; however, there would likely be a spatial component to the connectivity probabilities which we do not address here. We quantified the average remembered value relative to initial for 10 random initializations. Performance declines somewhat as connectivity decreases, but even networks with 10% connection probabilities can learn to store stimulus information (Figs. II.6 A, B).

Networks with initially stronger synaptic strengths have similar or even better performance with partial connectivity (Fig. II.16). Experiments show that working memory performance declines with age, which correlates with a reduction in number of synapses (Peters *et al.* (2008)).

II.4.5 Pre-training the Network

In the previous examples, each network is initialized with random connection weights. In reality, the working memory network will be continuously learning and will not start over with random connection weights when each new stimulus is presented. Consequently, we speculated that, once the network had learned to store one stimulus, it should be able to remember subsequently presented stimuli, even with minimal re-training. Related experimental work shows that performance in working memory tasks in children and young adults can be increased not only for trained tasks but for new tasks not part of the training: this coincides with strengthening of connectivity in the prefrontal cortex (Constantinidis and Klingberg (2016)).

To determine if our synaptic update rule enables the network to store new stimuli without further training, we first trained the networks (Eq. II.3) 1, 5 or 10 times. Each training event (or "trial") refers to the network learning to store information about a new stimulus for 3 seconds. Each new stimulus corresponded to a new random initialization of the firing rates $\vec{r}(t = 0)$, with corresponding initial representation value $\hat{s}(t = 0) = \sum_i d_i r_i(t = 0)$. We quantified the networks' abilities to represent these training stimuli, and found that the networks performed better on each subsequent stimulus: training improved performance (Fig. II.7 A). Next, we asked if after training on 1, 5, or 10 prior stimuli, the network could store information about a new stimulus without any more synaptic updates. We found that a network was able to store information about a new stimulus after being trained on at least 1 previous stimulus and did better after training on 5 or 10 stimuli (Fig. II.7 B).

II.4.6 Remembering Multiple Stimuli

The previous section shows how networks can self-organize to store information about one stimulus value, but working memory capacity in adult humans is typically 3–5 items (Cowan (2010)). To incorporate this working memory capacity into our models, we adapted the representation such that there were multiple read-out vectors, one for each stimulus value. We then derived plasticity rules via gradient descent on the squared and summed time derivatives of these representations: the loss was $\sum_k \left(\frac{d\hat{s}_k}{dt}\right)^2$. This led to the plasticity rule

$$\Delta L_{ij} = -\eta \sum_{k=1}^{n} \frac{d\hat{s}_k}{dt} d_{i_k} \frac{dr_i}{da_i}(t) r_j(t), \tag{II.4}$$

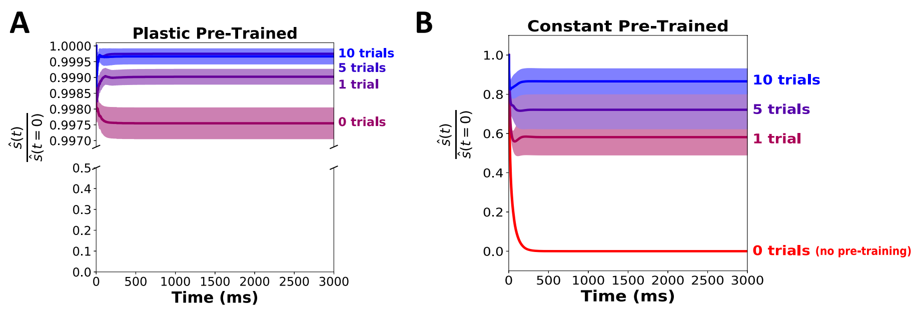


Figure II.7: Training improves performance. **(A)** The average remembered value relative to initial over 10 random initializations for a plastic random synapse network that has seen 0 previous stimuli, 1 previous stimulus, 5 previous stimuli or 10 previous stimuli. **(B)** The average remembered value relative to initial over 10 random initializations for a constant synapse network that has previously been trained on 0, 1, 5, or 10 previous stimuli, but with no training during the simulation period shown. Shaded areas in A and B represent ± standard error of the mean over the 10 different random initializations.

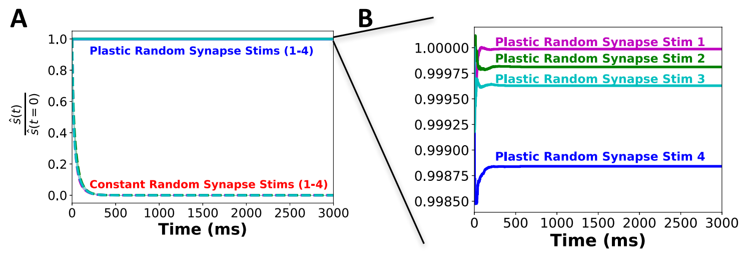


Figure II.8: Remembering four things. **(A)** The average remembered value relative to initial of four stimuli retained for 10 randomly initialized networks with constant random synapses (dashed lines) and plastic random synapses (solid lines). Different colors are for different stimulus values. The stimulus numbers (1-4) are arbitrarily labeled. Shaded areas represent ± standard error of the mean over the 10 different random initializations. **(B)**. A zoomed in look at the average remembered value relative to initial for the plastic model in panel A. Shading omitted from lines in panel B for clarity.

where n is the number of stimuli to be remembered. We chose $n = 4$ for our experiments, and we quantified how well the networks remember these stimuli (Stims 1-4 in Fig. II.8). In our experiments, each neuron in the network contributed to the representation of every stimulus value: *in vivo*, most neurons are sensitive to multiple aspects of stimuli (Miller and Fusi (2013)). This is not a requirement: the models successfully represent multiple stimuli even when different subsets of the neurons participate in each representation.

II.4.7 Potential Biological Mechanisms

Synaptic updates are thought to rely on synaptically local information, like the activities of the pre- and post-synaptic neurons. Our plasticity rules (Eqs. II.3, II.4) involve this information, in addition to "global" error value(s) $\{\frac{d\hat{s_k}d_i}{dt}\}$. Thus, we obtained three-factor rules: synaptic changes depend on the pre-

and post-synaptic neurons' activities, and a global error signal (Lillicrap *et al.* (2016)). There are two facets of the networks presented above that are unclear in their biological interpretations: the source (and precision) of the global error signal, and the symmetry of the feedback signals that convey the error information to the synapses. We discuss these issues in more detail below, and show that they are not firm requirements: our networks can learn to form memory representations even with more realistic asymmetric feedback, and with very coarse (even binary) error signals. Consequently, we demonstrate that synaptic plasticity mechanisms can successfully form self-organizing memory networks with minimal constraints.

II.4.7.1 *Sources of the Global Error Signal*

Below, we propose two sources for the global error signal(s), $\{\frac{d\hat{s_k}}{dt}\}$: feedback to the neurons' apical dendrites (Fig. II.9 A), and neuromodulatory chemicals that modulate the plasticity (Fig. II.9B). These are not mutually exclusive, and in either case, continuous training is not required for functioning working memory (Fig. II.7): the feedback signals need not be constantly available. In both implementations, a downstream "read-out" system estimates $\hat{s}$, and sends that information (or information about its time derivative) back to the memory network. This means that, at first glance, the read-out weights d_i must be accessed in two separate places: at the synapse, L_{ij}, to calculate the update, and at the read-out layer to calculate the remembered stimulus value (Eq. II.2). There are no known mechanisms that would allow this same d_i value to be accessed at distantly-located regions of the brain (Lillicrap *et al.* (2016)), posing a challenge to the biological plausibility of our networks. To address this issue, and show that known feedback and neuromodulatory mechanisms could implement our memory circuits, we later consider random (asymmetric) feedback weights, where the read-out weight used to estimate $\hat{s}$, differs from the one used to update the synaptic weights (Fig. II.9C). We also consider that error signals, either via segregated dendrites or neuromodulators, would likely be on a slower time scale than activity in the network. We show that the networks can self-organize with delayed plasticity (Fig. II.10). Networks with longer delays between plasticity updates require some pre-training before effectively storing information (Fig. II.10).

<u>Calculating the Global Error Signal Locally Using Segregated Dendrites:</u> The global error signal could be calculated locally by each neuron, by exploiting the fact that, in pyramidal cells, feedback arrives at the apical dendrites, and modulates synaptic plasticity at the basal dendrites (where information comes in from other cells in the memory network) (Lillicrap *et al.* (2016); Guergiuev *et al.* (2017)) (Fig. II.9 A). Here, a readout layer provides feedback to the apical dendrites that specifies the represented stimulus value $\hat{s}(t)$: the weight of the feedback synapse to neuron i from read-out neuron k is d_{i_k}, and so the apical

dendrite receives a signal $\sum_k d_{i_k} \hat{s}_k(t)$. The apical dendrites track the changes in these feedback signals, sending that information ($\sum_k d_{i_k} d\hat{s}_k/dt$) to basal dendrites via the soma. Correlating those signals with the pre- and post-synaptic activity at each of the synapses on the basal dendrites, the synaptic updates specified by Eq. II.4 are obtained. Thus, the neurons locally compute the synaptic updates.

Signalling the Global Error Signal with Neuromodulators: Alternatively, the global error signal(s) could be communicated throughout the network by neuromodulators, like dopamine, acetylcholine, serotonin, or norepinephrine. These have all been shown to be important in synaptic plasticity in the prefrontal cortex and in working memory (Meunier *et al.* (2017)). This is reward learning, with the reward values coming from the neuromodulator concentrations. Experimental work shows that synapses have activity-dependent "eligibility traces" that are converted into changes in synaptic strength by reward-linked neuromodulators (He *et al.* (2015)). In this scenario the concentration of different modulators tracks the error signals, $\frac{d\hat{s}_k}{dt}$, and the densities of the receptors to the modulators at each synapse are d_{i_k}. Thus, at each synapse, the modulators bring information $\sum_k d_{i_k} d\hat{s}_k/dt$ that, when correlated with the pre- and post-synaptic activities, yields the updates from Eq. II.4 (Fig. II.9 B).

II.4.7.2 *Learning with Fewer Constraints*

The discussion above shows that it is critical to implement our memory networks with asymmetric feedback weights. To do this, we let the top-down feedback impinge on the neurons at synapses with weights d_i, and the read-out layer calculate the remembered stimulus as:

$$\hat{s}(t) = \sum_i q_i r_i(t), \tag{II.5}$$

where $d \neq q$ (Lillicrap *et al.* (2016)). Here, the values of q_i are randomly drawn, independently from d_i, and so we refer to this as asymmetric random feedback.

To test whether networks with this asymmetric feedback could learn to store stimulus information, we simulated such networks and quantified the average remembered value relative to initial. The results (Fig. II.9 C, upper curve) show that even with asymmetric feedback, networks can learn to store stimulus information. This makes sense because both q and d contain only positive elements, so the feedback update signal to each synapse has the same sign as the update calculated from gradient descent. Thus, the synaptic updates with asymmetric feedback are generally in the same direction as those obtained from gradient descent (i.e., the angle between the update vectors, and those from true gradient descent, is less than 90^o), which suffices for learning (Lillicrap *et al.* (2016)).

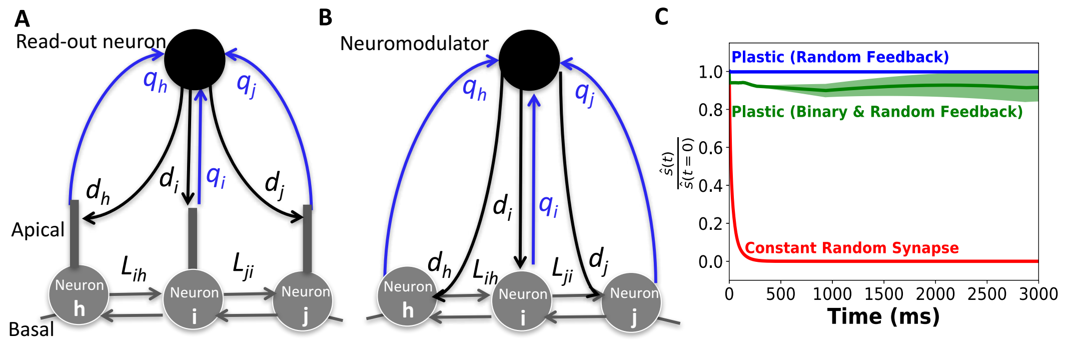


Figure II.9: Potential biological implementations. **(A)** Cartoon depicting error signals being calculated locally by the neurons' apical dendrites. The read-out neuron calculates the remembered stimulus value via Eq. II.5, where q_i is the weight of the synapses from cell i in the memory circuit to the read-out neuron. The apical dendrite calculates the error signal, weighted by d_i, by subtracting feedback values at adjacent time points. The basal dendrite receives inputs from neurons in the working memory circuit. The soma transmits the error signal to the basal dendrites, where the synaptic updates, L_{ij} are calculated by correlating the pre- and post-synaptic activities with the error signal. **(B)** Cartoon depicting error signals being communicated by neuromodulators. Neurons in the modulatory system calculate the remembered stimulus values via Eq. II.5, where q_i is the weight of the synapse from cell i in the memory circuit to the read-out neuron. That cell releases an amount of neuromodulator that tracks the changes in the represented stimulus value. The modulatory chemical affects synaptic plasticity by an amount that depends on the receptor density, d_i, at the synapses. Both implementations work with asymmetric feedback: the weights to the read-out neuron from cell i (q_i) will not necessarily match the weights, d_i, with which cell i receives the feedback signals. **(C)** The average remembered value relative to initial over 10 random initializations of a network with plastic random synapses with random feedback and readout weights ($q \neq d$) (blue), plastic random synapses with random feedback and readout weights and binary error signals ($\frac{d\hat{s}_k}{dt} \in \{\pm 1\}$) (green), and constant synapses (red). The shaded areas represent $\pm$ standard error of the mean over the 10 different random initializations.

Next, we wondered whether our networks require precise error signals $\frac{d\hat{s}_k}{dt}$ to learn to form memory representations, or whether coarser feedback would suffice: if coarser signals suffice, this removes any fine-tuning requirement “hidden" in the precision of the feedback. To answer this question, we repeated the simulations with our asymmetric random feedback networks, but binarized the error signals used in the synaptic plasticity, via the sign function: $\text{sign}(\frac{d\hat{s}_k}{dt}) \in \{\pm 1\}$. The updates are now reduced to either Hebbian or anti-Hebbian learning rules, depending on the sign of the error signal. The results (Fig. II.9 C, middle curve) show that with asymmetric and binary feedback, the networks can still learn to form memory representations, albeit not quite as well as in the case of highly precise feedback signals (Fig. II.9 C, upper curve). Networks with stronger initial synaptic strengths can store information with asymmetric and binary feedback as well (Fig. II.18). We found that networks with asymmetric and binary feedback were also robust to partial plasticity and partial connectivity (Fig. II.12).

Similar (anti-)Hebbian rules were used by Xie and Seung (2000) to tune an integrate-and-fire neuron with an excitatory autapse to maintain persistent activity under certain conditions. In comparison to that work, we studied more complex networks which do not rely on autapses for memory function. The Xie

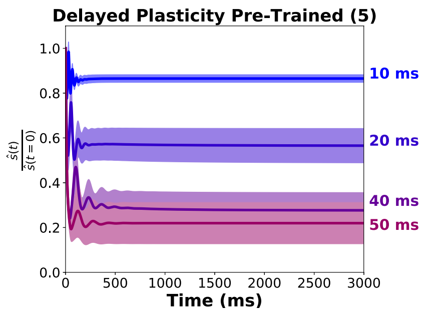


Figure II.10: Delayed plasticity. The average remembered value relative to initial over 10 random initializations of a plastic random network with delayed plasticity that been trained on 5 previous stimuli. The network initially has an anti-Hebbian learning rule, where error signal is -1 until feedback arrives after the delay period (10, 20, 40 or 50ms). The network then receives the error signal from time t minus the delay period. The shaded areas represent $\pm$ standard error of the mean over the 10 different random initializations.

and Seung (2000) model also requires some fine-tuning for the neuron to continue to store information about a stimulus (Seung *et al.* (2000)).

II.4.7.3 *Delayed Feedback*

Based on the potential sources for the global error signal(s) (Figs. II.9 A, B), there could be a delay before the feedback signal would reach the working memory network. To test whether our network could still learn to store information even if the error signals were delayed, we simulated networks with initially anti-Hebbian learning rules (error signal is -1) until some delay period, (10-50ms), and then the network receives the error signal from time t minus the delay period. Because this is a difficult task, we lowered the learning rate, η, and allowed the network to train on 5 individual previous stimuli, with the delayed plasticity, and then evaluated its ability to store information (while still training) on a novel stimulus. We found that networks with delays in feedback could still store information (Fig. II.10). Even with delays of 40-50ms the networks could still retain some information (Fig. II.10).

II.5 Additional Results

II.5.1 Larger Networks

We performed experiments with 100 neurons to make repeated tests more efficient, but to ensure our network set-up would be able to scale up we also implement our learning rule in networks with 1000 and 10,000 neurons (Fig. II.11) and demonstrate that these networks are able to store information.

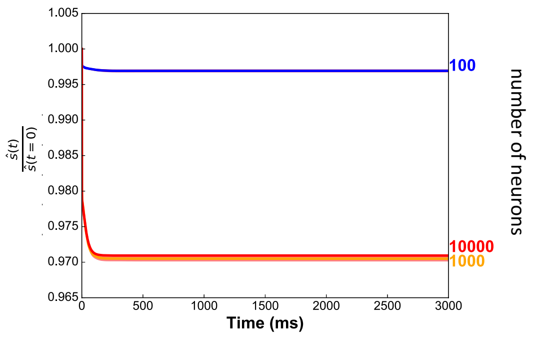


Figure II.11: The average remembered value relative to initial for 10 random initializations of networks with 100 neurons (blue), 1000 neurons (yellow) and 10,000 neurons (red). The only differences between the networks aside from size are the learning rates, which should be reduced as the network size increases.

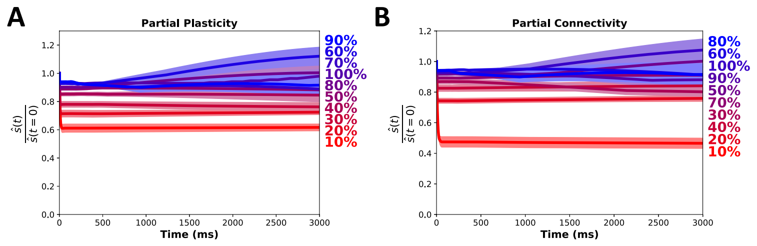


Figure II.12: Partial plasticity and connectivity in networks with asymmetric random feedback and binary error signals. (Similar to Figs. II.5 and II.6, but with binary error signals and asymmetric random feedback, as in Fig. II.9) **(A)** The average remembered value relative to initial for 10 random initializations of plastic random synapse networks with asymmetric random feedback and binary error signals. Different lines are for different percentages of plasticity. **(B)** The average remembered value relative to initial for 10 random initializations of plastic random synapse networks with random feedback and readout weights and binary error signals. Different lines are for different connection probabilities. Shaded areas in A and B represent ± standard error of the mean over the 10 different random initializations.

II.5.2 Robustness Tests

We demonstrated that with asymmetric and binary feedback, the networks can still learn to form memory representations (Fig. II.9 C, upper curve). We then tested whether networks with random and imprecise feedback are still robust to partial plasticity and partial connectivity. We followed the same protocol as sections II.3.1, but in networks with asymmetric and binary feedback (as in section II.4.7.2) and found that networks were still able to store information even with just 10% plasticity or 10% connectivity, albeit not as well as in the case of highly precise feedback signals (Fig. II.12).

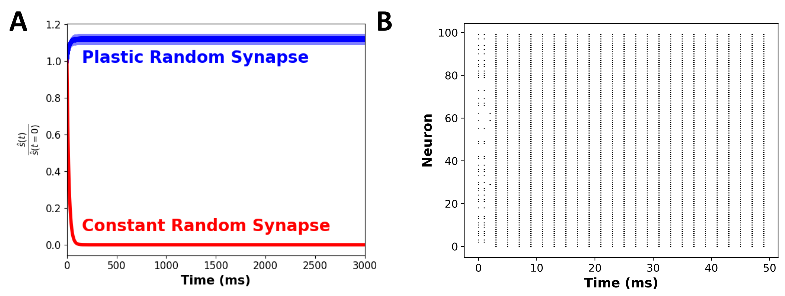


Figure II.13: Stimulus retention in leaky integrate and fire (LIF) spiking self-organizing memory networks which enforce Dale's Law (neurons are either excitatory or inhibitory). **(A)** The average remembered value relative to the initial stimulus value over 10 random initializations for the plastic random synapse model (blue) and the constant random synapse model (red) where 1.0 indicates perfect memory. Shaded areas represent $\pm$ standard error of the mean over the 10 different random initializations. **(B)** Raster plot of spiking activity of the excitatory neurons which encode for the remembered stimulus value over 50ms of one random initialization.

II.5.3 Stimulus Retention in LIF Networks

In order to test if our update rule can organize a spiking network, we implemented a leaky integrate-and-fire (LIF) model (Ledoux and Brunel (2011)). We enforced Dale's law with separate E and I populations composed of $N_E = 100$ and $N_I = 20$ LIF neurons. Each excitatory (E) neuron i was described by its internal activity (membrane potential) $a_i(t)$ which obeys:

$$\tau_E \frac{da_i(t)}{dt} = a_{rest} - a_i(t) + \sum_{j \in E} L_{EE_{ij}} m_{ij}(t) - \sum_{j \in I} L_{EI_{ij}} m_{ij}(t) \tag{II.6}$$

where τ_E is the membrane time constant of E neurons (20ms), a_{rest} = -70mV is the resting membrane potential, L the synaptic strength, and $m_{ij}(t)$ are individual synaptic currents modeled with an additional synaptic variable x_{ij}:

$$\tau \frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \tau \sum_k \delta(t - t_j^k) \tag{II.7}$$

$$\tau \frac{dm_{ij}(t)}{dt} = -m_{ij}(t) + x_{ij}(t) \tag{II.8}$$

where τ is the time constant, the sum over k represents a sum over all spikes of pre-synaptic neuron j, that occurs at time t_j^k. Similarly, each inhibitory (I) neuron i was described by its internal activity (membrane potential) $a_i(t)$ which obeys:

$$\tau_I \frac{da_i(t)}{dt} = a_{rest} - a_i(t) + \sum_{j \in E} L_{IE_{ij}} m_{ij}(t) - \sum_{j \in I} L_{II_{ij}} m_{ij}(t) \tag{II.9}$$

where τ_I is the membrane time constant of I neurons (10ms), and the rest of the variables are defined as above for Eq. II.4. In this model, action potentials occurred when voltage crosses $V_{thr} = -50mV$ and at the time of the action potential, the voltage was reset at $V_{reset} = -60mV$, with no refractory period for simplicity (Ledoux and Brunel (2011)). In order to calculate the updates to the synapse, we took a smoothed rolling estimate of the firing rates $r_{E,I}(t)$ which evolved via:

$$\tau \frac{dr(t)}{dt} = -r(t) + \sum_i \delta_i(t), \tag{II.10}$$

where $\delta_i(t)$ is a binary variable for if neuron i spikes or not. In order to organize the network to store information, we updated the synapses of the excitatory network, L_{EE} with the same derived update as Eq. II.3 except the loss was calculated as:

$$\tau \frac{d\hat{s}}{dt} = -\hat{s}(t) + \sum_{i \in E} d_i \delta_i(t), \tag{II.11}$$

where $\hat{s}(t)$ decays exponentially to 0 without spikes $\delta_i(t)$. To evaluate the performance of our spiking working memory networks, we asked how well the networks could store information about a scalar stimulus value, as in sec. II.3.1. The networks were all-to-all connected, autapses excluded (so diagonal elements of L_{ij} are all zero) and contain 100 excitatory neurons and 20 inhibitory neurons. We evaluated how well random networks without plastic synapses store information, by initializing each network with random neural activities, $\vec{a}(t = 0)$, random read-out weights, $\vec{d}$, and random connection weight matrices, L, and simulating the dynamical evolution of the representation. The initial stimulus value to be remembered is a weighted combination of the spikes from the excitatory population, $\hat{s}(t = 0) = \sum_i \delta_i(t = 0) d_i$. We found that LIF networks with plastic synapses updated via Eq. II.3 are able to store information about stimulus (Fig. II.13 A). The network learns to maintain information with periodic bursts of spikes (Fig. II.13 B).

II.5.4 Initially Stronger Synaptic Networks

In Figure II.2 we demonstrate that the network learns to store information with initially weaker or stronger initial synaptic weights. In networks with initially weaker synaptic weights, the networks learns to stabilize activities, whereas in networks with initially stronger synaptic weights the neural activities are time-varying. We repeated the experiments of the main results in networks with initially stronger synaptic weights to compare robustness. The following networks are initialized with stronger synaptic weights where elements of L_{ij} were drawn from Gaussian distribution mean 0 standard deviation $\frac{1.6}{\sqrt{N}}$ (the network is initially in a chaotic state, as discussed by Toyoizumi and Abbott (2011)).

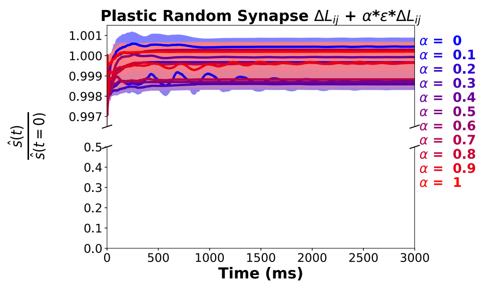


Figure II.14: (Similar to Fig. II.4 B, but with stronger initial synaptic weights). The average remembered value relative to initial for 10 random initializations of networks with differing levels of synaptic update noise ($\alpha \leq 1$) in networks with initially stronger synaptic connections. Shaded areas represent $\pm$ standard error of the mean over the 10 different random initializations.

II.5.4.1 *Noisy Synapses*

We added Gaussian noise with mean 0 and standard deviation α times the update to the synapse, ΔL_{ij}, where α was varied from 0 to 1 as in section II.4.3.1. We quantified the average remembered value relative to initial for networks with various noise levels and found that the noise did not have a noticeable effect on the network performance for $\alpha \leq 1$ in networks with initially stronger synaptic weights (Fig. II.14).

II.5.4.2 *Partial Tuning*

To determine how well the network performs if only some synapses are updated, we simulated networks in which different fractions of the synapses were updated using Eq. II.3: the other synapses were held constant. We quantified the remembered value relative to initial for these networks (Fig. II.15). As found in Fig. II.5, even with just 10% of the synapses being tuned, the networks with initially stronger synaptic weights learn to store information about the stimulus.

II.5.4.3 *Partial Connectivity*

To ask if our synaptic update rule could self-organize partially connected networks, we simulated networks with different connection probabilities and with synapses updated using Eq. II.3. In networks with initially weaker synaptic strengths, 1% connection probability led to complete loss of memory (Fig. II.6 A). In networks with initially stronger synaptic strengths, 1% connection probability was sufficient for storing some information about the stimulus (Fig. II.15). As in sec. II.4.4, we found that performance declines somewhat as connectivity decreases, but even networks with 10% connection probabilities can learn to store stimulus information (Fig. II.16).

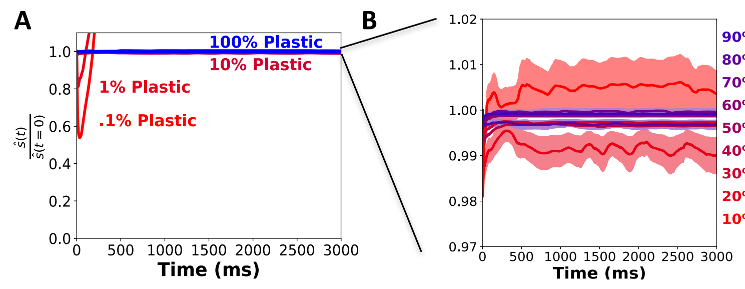


Figure II.15: Robustness to partial plasticity in networks with initially stronger synaptic weights. (Similar to II.5, but with stronger initial synaptic weights). **(A)** The average remembered value relative to initial for 10 random initializations. Different lines are for different fractions of plastic synapses. **(B)** A zoomed in look at the average remembered value relative to initial for $10\% - 90\%$ plasticity. Shaded areas in B represent $\pm$ standard error of the mean over the 10 different random initializations. Shading omitted from lines in panel A for clarity.

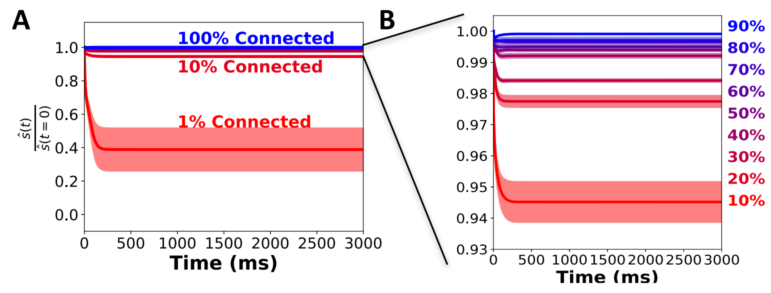


Figure II.16: Network robustness to partial connectivity. (Similar to Fig. II.6, but with stronger initial synaptic weights). **(A)** The average remembered value relative to initial for 10 random initializations. Different lines are for different connection probabilities. **(B)** A zoomed in look at the average remembered value relative to initial for $10\% - 90\%$ connection probability. Shaded areas in A and B represent $\pm$ standard error of the mean over the 10 different random initializations

II.5.4.4 *Pre-training the Network*

We then determined if our synaptic update rule enables the network with stronger initial synaptic strengths to store new stimuli without further training. We first trained the networks (Eq. II.3) 1, 5 or 10 times. Each training event (or “trial") refers to the network learning to store information about a new stimulus for 3 seconds. Each new stimulus corresponded to a new random initialization of the firing rates $\vec{r}(t=0)$, with corresponding initial representation value $\hat{s}(t=0) = \sum_i d_i r_i(t=0)$. We quantified the networks' abilities to represent these training stimuli, and found less clear improvement on subsequent stimulus as we found in networks with initially weaker synaptic strengths (Fig. II.7 A, Fig. II.17 A). Next, we asked if after training on prior stimuli, the network could store information about a new stimulus without any more synaptic updates. While networks with initially weaker synaptic strengths are able to store information about a new stimulus after being trained on at least 1 previous stimulus (Fig. II.7 B),

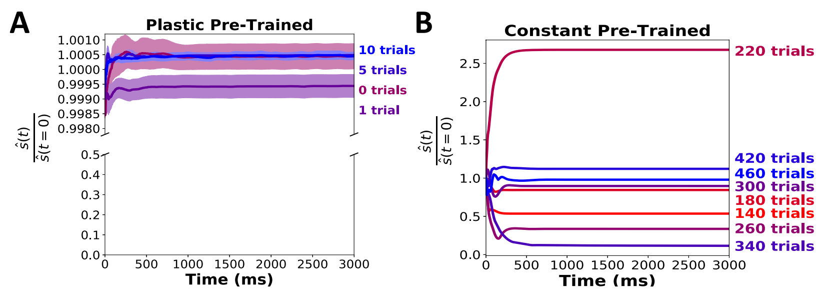


Figure II.17: Performance over more training. (Similar to Fig. II.7, but with initially stronger synaptic weights). **(A)** The average remembered value relative to initial over 10 random initializations for a plastic random synapse network that has seen 0 previous stimuli, 1 previous stimulus, 5 previous stimuli or 10 previous stimuli. **(B)** The average remembered value relative to initial for one random initialization for a constant synapse network that has been previously trained on 140 to 460 previous stimuli, but with no training during the simulation period shown. Shaded areas in A represents ± standard error of the mean over the 10 different random initializations.

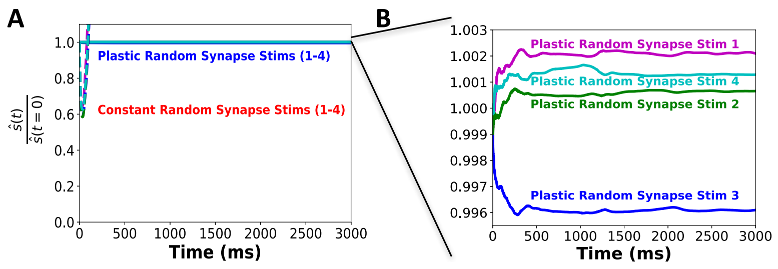


Figure II.18: Remembering four things. (Similar to Fig. II.8, but with initially stronger synaptic weights). **(A)** The average remembered value relative to initial of four stimuli retained for 10 randomly initialized networks with constant random synapses (dashed lines) and plastic random synapses (solid lines). Different colors are for different stimulus values. The stimulus numbers (1-4) are arbitrary labeled. **(B)**. A zoomed in look at the average remembered value relative to initial for the plastic model in panel A. Shading omitted from lines for clarity.

we found that significantly more trials are required for networks with stronger initial synaptic strengths, and there is still a lot of variability even after hundreds of trials (Fig. II.17 B).

II.5.4.5 *Remembering Multiple Stimuli*

We quantified how well our networks remember 4 stimuli as in sec. II.4.6, with the learning rule from Eq. II.4, and found that networks with initially stronger synaptic weights can store multiple stimuli (Fig. II.18).

II.5.4.6 *Learning with Fewer Constraints*

We found that networks with initially stronger synaptic strengths could learn with asymmetric feedback where we calculate the remembered stimulus as Eq. II.5 (Fig. II.19, upper curve). We also found that the networks were able to store information with asymmetric feedback and binarized error

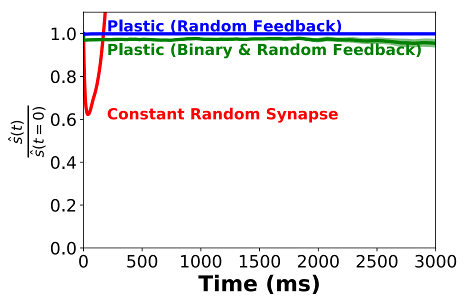


Figure II.19: (Similar to Fig. II.9 C, but with initially stronger synaptic weights). The average remembered value relative to initial over 10 random initializations of a network with plastic random synapses with random feedback and readout weights ($q \neq d$) (blue), plastic random synapses with random feedback and readout weights and binary error signals ($\frac{d\hat{s}_k}{dt} \in \{\pm 1\}$) (green), and constant synapses (red). The shaded areas represent $\pm$ standard error of the mean over the 10 different random initializations. Shading omitted from 'Constant Random Synapse' (red) for clarity

signals as described in sec. II.4.7.2 (Fig. II.19, middle curve) with higher performance than networks with initially weaker synaptic strengths (Fig. II.9 C, middle curve).

II.6 Future Work

We have demonstrated that an RNN can be trained with biologically plausible plasticity rules to represent stimuli required for short-term memory via three-factor rules: synaptic changes depend on the pre- and post-synaptic neuron's activities, and a global error signal. Potential sources for this global error signals include neuromodulators - a chemical signal that regulates a population neurons - and via local calculation on the apical dendrite of a neuron which receives feedback information from other neurons. We propose important future work would be training RNNs with biologically plausible three-factor rules for more brain-like learning and better training efficiency. This will provide a more efficient and biologically relevant machine learning tool which will be contributed to open source for repeating experiments as well as being applied to alternative problems.

RNNs are a valuable tool for learning time-varying inputs and performing time series forecasting (Sussillo and Abbott (2009); Prasad and Prasad (2014)) because of their memory of former inputs. While RNNs are based on biologically realistic connectivity found in cortex (Schmidhuber (1993)), backpropagation, the commonly used algorithm for training RNNs, is not. Backpropagation (backprop) evaluates error by multiplying error signals with each synaptic weight on every neuron so that each neuron is properly updated based on its contribution to the error. This involves a precise, symmetric connectivity pattern which is thought to be impossible in the brain (Lillicrap *et al.* (2016); Rumelhart *et al.* (1986b); Grossberg (1987)). Further, each synapse needs to be able to look back at the entire network throughout time (the weight assignment problem), which is highly unlikely to be implemented by the brain (Lillicrap *et al.* (2016); Crick (1989); Grossberg (1987)).

Similar learning rules to ours have successfully been utilized to train a feedforward network to recognize handwritten digits (MNIST dataset) (Lillicrap *et al.* (2016); Guergiuev *et al.* (2017)). There has been some work demonstrating the use of biologically plausible trained RNNs (Sussillo and Abbott (2009); Alemi *et al.* (2015)), yet the use of biologically plausible three-factor rules to train RNNs has not been sufficiently explored.

II.7 Discussion

We derived biologically plausible synaptic plasticity rules through which networks self-organize to store information in working memory. Networks implementing these plasticity rules are robust to synaptic noise, to having only some of the synapses updated, and to partial connectivity. These networks can store multiple stimuli and have increased performance after previous training. We suggest two candidate sources for the global error signal necessary for the plasticity rule, and demonstrate that our networks can learn to store stimulus information while satisfying the added requirements imposed by these biological mechanisms. This flexibility suggests that other types of synaptic plasticity updates may also be able to organize working memory circuits.

Prior work described storing time-varying inputs, which is more complex than the working memory task we consider here, where the network must store a static input (Bourdoukan *et al.* (2012); Bourdoukan and Deneve (2015); Deneve *et al.* (2017); Brendel *et al.* (2017)). The networks in these prior studies have two different synapse types ("fast" and "slow"), and their slow synapses used learning rules mathematically similar to our Eq. II.3, but with error signals that depend on externally-defined target signals. The prior studies, however, have not addressed the robustness of networks implementing these learning rules to factors like imprecise synaptic weights and updates, partial connectivity (e.g., not all-to-all), asymmetric connectivity, and plasticity that affects only some synapses. For contrast, we demonstrate that networks implementing our plasticity rule *are* robust to the factors listed above. It will be interesting for future work to study the robustness of synaptic plasticity in networks that perform more complex tasks. This would clarify if the robustness we observed in our networks is from the learning rule itself, or from the low dimensionality of the signal in our simple working memory task.

A potential caveat in using a rate-based network model is losing information about spike-timing dependency. We tested our plasticity rules in leaky integrate and fire models and found that spiking networks implementing these update rules learn to store information (Fig. II.13). We also enforced Dale's law in this model with separate excitatory and inhibitory populations of neurons. We used a similar learning rule as the rate-based model, but with a smoothed rolling estimate of the firing rates. We found that in this model, the network maintains information about the stimulus with periodic bursts of spikes.

We found that the network learns to store information by stabilizing the activities. In the model presented by Druckmann and Chklovskii (2012), however, the neural activity was time-varying while the network was still able to store information. We repeated our experiments with networks that started with initially stronger synaptic weights and demonstrate that the neural activities can be time-varying in our network and still store information about the stimulus.

Along with understanding how information is stored in working memory, this work may have implications in machine learning technology. This work uses the idea of Reservoir Computing, where the network consists of a reservoir, or internal part, with interconnected neurons that deliver an output to a readout layer (Soriano (2017); Marzen (2017); Dambre *et al.* (2012); Inubushi and Yoshimura (2017)). Reservoir computing has shown excellent performance in processing tasks such as time series forecasting and speech recognition, because of its ability to store memory and predict future events (Soriano (2017); Marzen (2017); Dambre *et al.* (2012); Inubushi and Yoshimura (2017)).

Machine learning algorithms are often unrealistic from a biological perspective: most rely on non-local synaptic updates or symmetric synapses. We show that recurrent networks can learn to store information using biologically plausible synaptic plasticity rules which require local information plus a global error signal (or signals), that can be calculated on the apical dendrite or via neuromodulators. This same setup could be utilized in RNNs to make them more biologically realistic. This would let us better understand how the brain learns, and could lead to novel biomimetic technologies: prior work on biologically realistic machine learning algorithms has led to hardware devices that use on-chip learning (Knag *et al.* (2015); Zylberberg *et al.* (2011)). Synaptically local updates do not have to be coordinated over all parts of the chip, enabling simpler and more efficient hardware implementations. We demonstrate preliminary work in RNNs that can learn to follow chaotic time series. There is still a lot of work to be done in biologically plausible RNNs, for example our networks are unable to continue to produce chaotic patterns reliably once we turn off training.

CHAPTER III
TRAINING NEURAL NETWORKS WITH BRAIN DATA IMPROVES OBJECT RECOGNITION PERFORMANCE

III.1 Summary

The current state-of-the-art object recognition algorithms, deep convolutional neural networks (DCNNs), are inspired by the architecture of the mammalian visual system, and capable of human-level performance on many tasks. However, even these algorithms make errors. As DCNNs improve at object recognition tasks, they develop representations in their hidden layers that become more similar to those observed in the mammalian brains. Moreover, DCNNs trained on object recognition tasks are currently among the best models we have of the mammalian visual system. This led us to hypothesize that teaching DCNNs to achieve even more brain-like representations could improve their performance. To test this, we trained DCNNs on a composite task, wherein networks were trained to: a) classify images of objects; while b) having intermediate representations that resemble those observed in neural recordings from monkey visual cortex. Compared with DCNNs trained purely for object categorization, DCNNs trained on the composite task had better object recognition performance on held out data, indicating that using neural data in the training procedure could lead to advances in object recognition systems. Further, the networks trained on the composite task better match the responses of visual cortical neurons to natural images. Our results outline a new way to regularize object recognition networks, using transfer learning strategies in which the brain serves as a teacher for training DCNNs.

III.2 Introduction

Deep convolutional neural networks (DCNNs) have recently led to a rapid advance in the state-of-the-art object recognition systems (Lecun *et al.* (2015)). At the same time, there remain critical shortcomings in these systems (Rajalingham *et al.* (2018)). For example: DCNNs are much more sensitive to image manipulations like gray-scaling than are humans (Geirhos *et al.* (2018)); DCNNs have a strong bias towards textural information over shape (Geirhos *et al.* (2019)); and DCNNs are subject to adversarial examples (small pixel-level image manipulations, imperceptible to a human, that nevertheless cause the DCNN to change the category label it assigns to the image (Goodfellow *et al.* (2015)). Given these challenges, we asked whether training DCNNs to respond to images in a more brain-like manner could lead to better performance. Motivating us, DCNN architectures are directly inspired by that of the mammalian visual system (MVS) (Hubel and Wiesel (1968)), and as DCNNs improve at object recognition tasks, they learn representations that are increasingly similar to those found in the MVS

(Kriegeskorte (2014); Yamins and Dicarlo (2016); Gu and van Gerven (2015); Mcclure and Kriegeskorte (2016)). Consequently, we expected that forcing the DCNNs to have image representations that were *even more* similar to those found in the MVS, could lead to better performance.

Previous work showed that the performance of smaller "student" DCNNs could be improved by training them to match the image representations of larger "teacher" DCNNs (Mcclure and Kriegeskorte (2016); Romero *et al.* (2015); Hinton *et al.* (2015)), and that DCNNs could be directly trained to reproduce image representations formed by the V1 area of monkey visual cortex (Kindel *et al.* (2019)). These studies provide a foundation for the current work, in which we used monkey V1 as a teacher network for training DCNNs to categorize images. This is a form of transfer learning, where we used the monkey brain to constrain the image representations within the DCNN. DCNNs trained with the monkey V1 as a teacher outperformed those trained without this teacher signal, by several relevant metrics. We did not aim to achieve state-of-the-art classification performance, as we are studying neural networks that are small relative to the current state-of-the-art (the largest networks we considered have the VGG-16 architecture). However, our results indicate that, even for moderately powerful DCNNs, performance can be improved by using data from the monkey brain as a teacher signal. We anticipate that future studies could apply this training method to larger networks, thereby improving on the current current state-of-the-art object recognition systems.

III.3 Methods

III.3.1 Monkey Visual Cortex Data

We used publicly-available multielectrode recordings from anesthetized monkeys presented with a series of images while experimenters recorded the spiking activity of neurons in primary visual cortex (V1) with a multielectrode array (Coen-Cagli *et al.* (2015)). These recordings were conducted in 10 experimental sessions with 3 different animals, resulting in recordings from 392 neurons. In addition to these V1 data, which is our main focus, we also studied recordings from cortical areas V4 and IT (Majaj *et al.* (2015)) (sec. III.4.12). Similar to the V1 data, these recordings were performed with multielectrode arrays implanted in the cortex.

III.3.2 Representational Similarity Matrices (RSMs)

To compare image representations in the monkey brain with those in a DCNN, we used representational similarity matrices (RSMs) (Kriegeskorte *et al.* (2008)). For each pair of images (i & j) shown to the monkey, we computed the similarity between measured neural responses (v_i & v_j): these vectors contain the firing rates of all of the observed neurons. We assessed the similarity by the cosine similarity

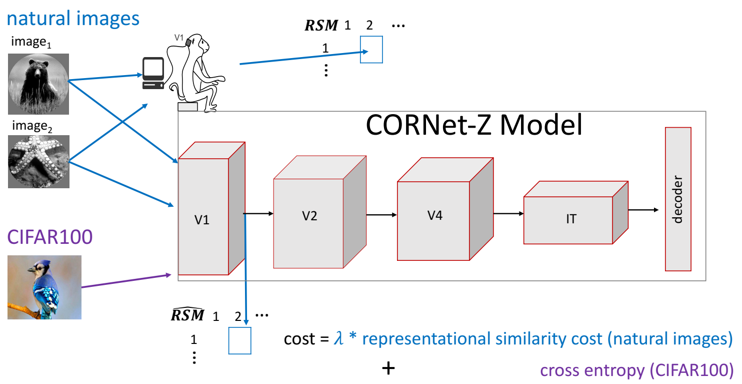


Figure III.1: CORNet-Z has multiple blocks, each of which consists of a convolution followed by a ReLU nonlinearity and max pooling. The blocks are identified with cortical areas V1, V2, V4 and IT, which exist at the corresponding depths in primate visual system. The cost function is a weighted combination of classification error (cross entropy) and representational similarity, weighted by λ.

between those vectors (Fig. III.1). These values (RSM_{ij}) were assembled into matrices, describing the representational similarities (Kriegeskorte *et al.* (2008)) in monkey V1, for all image pairs (i & j). We averaged the representational similarity matrices over the 10 experiments to yield a single RSM that was used for training the neural networks. During DCNN training, we input the same pairs of images (i & j) into our DCNNs as were displayed to the monkeys, and computed representational similarity matrices for the chosen layer of hidden units (Fig. III.1): the DCNN's RSM is denoted by $\widehat{RSM}$.

III.3.3 Deep Convolutional Neural Networks and Cost Functions

We performed most of our experiments on the CORNet-Z DCNN architecture, described below. We also performed some experiments on the larger VGG-16 architecture. Results were similar for both architectures. The CORNet-Z DCNN architecture (Kubilius *et al.* (2018)) is a trimmed-down version of the AlexNet (Krizhevsky and Hinton (2015)) object recognition algorithm (Fig. III.1). The layers in CORNet-Z have been identified with the brain areas at the corresponding depths within the mammalian visual hierarchy (Kubilius *et al.* (2018)) (Fig. III.1).

We trained the networks on the CIFAR100 (Krizhevsky (2009)) task, which consists of classifying images of objects from 100 different categories. Regardless of the network architecture, we randomly initialized all weights, and trained the DCNN to minimize a cost function consisting of two terms: classification error, and mismatch between the network's hidden representations and those in monkey V1. Classification error was computed as the cross entropy between the network's final outputs and the true object labels. Representation mismatch was computed as the mean-squared error between the monkey V1 representational similarity matrix (RSM), and that of the relevant layer of the DCNN ($\widehat{RSM}$).

For CORNet-Z, this was the V1 block (Fig. III.1), and for VGG-16, this was the third convolutional layer. That layer was chosen because, in VGG-16 networks trained for object recognition tasks, it has representations most similar to those seen in monkey V1 (Cadena *et al.* (2017)).

A trade-off parameter, λ, determines the relative weighting of the two terms in the cost function

$$cost = \lambda \sum_{i,j} (RSM_{ij} - \widehat{RSM}_{ij})^2 - \sum_i \hat{y_i} log(y_i). \tag{III.1}$$

We updated the trade-off parameter λ throughout training, so as to keep the ratio between the two terms in the loss function constant. In other words, λ was updated so that r was constant, with $r = \lambda \left[\sum_{i,j}(RSM_{ij} - \widehat{RSM}_{ij})^2\right] / [\sum_i \hat{y_i} log(y_i)]$. We studied networks with several different values of this ratio, r. We also experimented with using a constant λ throughout training but found this constant-ratio method leads to better object recognition performance (sec. III.4.10).

III.3.4 Training Procedures

We trained each CORNet-Z network for 100 epochs (250 for VGG-16). For most of our experiments, networks trained with neural data (i.e., with $r > 0$), were trained to minimize the composite cost (Eq. III.1) for the first 10 epochs, and thereafter were trained on just the cross entropy loss (this increased from 10 to 100 epochs for the VGG-16 experiments). In other words, we set $r \to 0$ after these first 10 epochs. This procedure reduces the computational cost due to evaluating the representational similarity matrices. We performed some experiments in which the neural data regularizer was applied at all training epochs, and saw similar results (sec. III.4.8).

We kept a static training rate of 0.01 for all networks and a batch size of 128 for CORNet-Z or 256 for VGG-16. We used dropout regularization (Srivastava *et al.* (2014)) with 0.5 retention probability for the 3 fully connected layers of all networks. For the training images, we centered the pixels globally across channels. The held-out testing images were not preprocessed, nor were the natural images that were presented to the monkeys in the neuroscience experiments. The trained network weights, and code associated with this chapter, can be downloaded at github.com/cfederer/TrainCNNsWithNeuralData.

For each architecture and choice of parameters, we repeated the training from 10 different random initial conditions; results reported are mean $\pm$ SEM. This approach has a larger computational cost than does reporting the result of a single training run, but makes it more likely that our findings will generalize because they do not depend on the idiosyncrasies of weight initializations.

III.3.5 Randomized RSMs

For control experiments, to test whether monkey V1 RSMs confer some benefit in neural network training, above that which could be obtained from randomly-drawn RSMs (e.g., to test whether arbitrary RSM constraints were just as good), we repeated our experiments with randomly-generated RSMs in place of the monkey V1 RSMs. We generated the random RSMs in several different ways.

First, we drew 39-element i.i.d. random vectors from a Gaussian distribution with $\mu = 5$ and $\sigma = 0.582$; different random vectors were drawn for each image. The number of elements (39) matches the average number of neurons simultaneously observed in one of our neuroscience experiments, and the variance (0.582) matches the variance of the measured neural data. The mean of these randomly-generated vectors differs from the neural data; we later consider random data with the same mean as in the V1 dataset. Given these random vectors, we then calculated the representational similarity matrices from the randomly drawn vectors. In Figs. III.5 and III.7, that RSM is dubbed “random".

To test whether having ‘V1-like’ data would suffice for the training, we drew 39-element vectors from i.i.d. Gaussian distributions with the same mean and variance as were seen in the monkey data: $\mu = 0.495$ and σ=0.582. We then used these random vectors to generate a RSM, as described above. This RSM was dubbed “random (V1-stats)".

Finally, we applied a shuffling procedure to the V1 data, where we randomly permuted the image labels associated with each recorded vector of neural firing rates. As a result of this procedure, the neural responses no longer matched the images that were shown to the monkey. The random permutation was done independently for each neuron. This leads to vectors of firing rates that match (for each neuron, and to all orders) the distributions seen in the monkey data, but removes information about the specific image features those neurons represent (i.e., the neurons’ receptive fields). Similar to the above experiments, we assembled these vectors into a RSM. This is referred to as “V1 shuffled".

III.4 Results

We trained neural networks on the composite cost (Eq. III.1), with varying ratios r describing the trade-off between representational similarity cost and categorization cost. We evaluated the trained networks based on categorization accuracy achieved on held-out data (not used in training) from the CIFAR100 dataset. In our figures, black lines indicate networks trained purely for categorization ($r = 0$), while red lines indicate networks trained using monkey V1 as a teacher ($r > 0$; see Methods). Higher weightings of neural data in the loss function correspond to the shade of red lines.

III.4.1 Improved Accuracy with Neural Data

We first present results from the CORNet-Z architecture, and we discuss results from the larger VGG-16 model in sec. III.4.13. We tested the trained models' ability to classify previously-unseen images from the CIFAR100 dataset (i.e., images not used in training), and quantified the fraction of images correctly labeled. Networks trained to both classify objects and match neural representations (i.e., those with $r > 0$) had better object recognition performance (Fig. III.2).

III.4.2 Model of V1

We tested how well the different networks performed as models of the visual system. To do this, we trained networks with different waiting ratios, r, on one of the 10 recording sessions from (Coen-Cagli *et al.* (2015)). We then computed the mean-squared error between the models' representational similarity matrices and monkey V1 similarity matrices from a different session recorded from a different monkey. This analysis revealed that networks trained on the composite task of classification and matching representational similarities to neural data form better models of monkey V1 (Fig. III.3).

III.4.3 Unit Activations

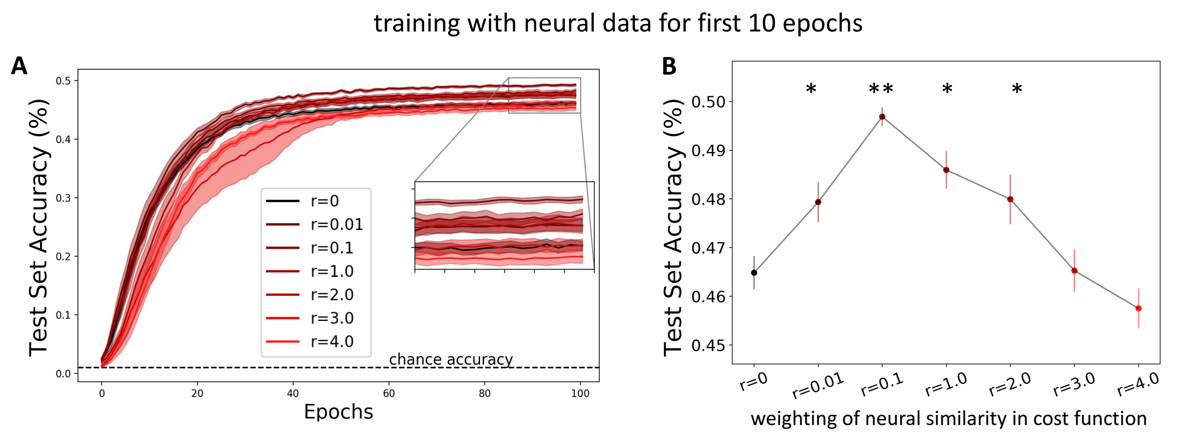


Figure III.2: Accuracy in categorizing previously-unseen CIFAR100 images for the CORNet-Z architecture trained with different weighting ratios, r, applied to monkey V1 representation similarity for the first 10 epochs of training. A) Test-set accuracy at each epoch during training. Chance accuracy indicated by dashed black line. Shaded areas are +/- SEM over 10 different random initializations of each model. B) Test-set accuracy for previously-unseen CIFAR100 images at the end of training, as a function of the weight (r) given to neural representational similarity in the cost function. Double asterisks (**) indicate significantly different results from no neural data, $r = 0$, at p<.001 on a one-tailed t-test. Single asterisks (*) indicate significantly different results from no neural data, $r = 0$, at p<.05 on a one-tailed t-test.

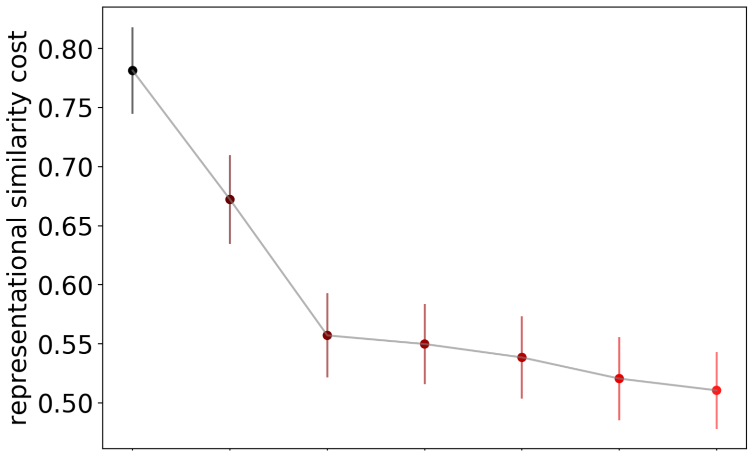


Figure III.3: Regularizing the network with data from one session and then testing on another session from a separate monkey shows that we can get a better model of V1 by regularizing with brain data. Error bars are +/- SEM over all batches of testing data.

Why might the networks trained to mimic the monkey brain have better object recognition performance? To gain some insights into this question, we used t-SNE (van der Maaten and Hinton (2008)) to visualize the unit activations from the first convolutional layer of the networks from the preceding section (i.e., the layer of CORNet-Z that aligns with monkey V1 in terms of depth in the visual pathway). We input images from the CIFAR100 test set into the networks, and used t-SNE to embed those data into two dimensions. We repeated this procedure for networks trained with different ratios r that dictate the trade-off between representational similarity cost and categorization cost.

While the representations of images from different categories remain co-mingled at this low level of the neural network, the representations are more varied for the network trained with a neural representation weighting of $r = 0.1$ (Fig. III.4 B) than for the network trained no neural data (r=0: Fig. III.4 A). This motivated us to compute the average variance per unit within the V1-like layer of the CORNetZ networks (variance in activations over the test-set images, averaged over all units in that hidden layer). As the weighting ratio r of neural similarity in the cost function increases, so does the activation variance (Fig. III.4 C). While this increase in activation variance is initially associated with increasing object-recognition performance (e.g., up to $r = 0.1$), at higher values of r, that increased unit activation variance no longer correlates with higher accuracy (e.g., for r=3.0 or r=4.0: Fig. III.4 C). These data suggest that the improved generalization performance obtained by using neural data in the training procedure (i.e., Fig. III.2) could arise in part because the training procedure that uses neural data forces the networks to have more diverse activations in their low-level units.

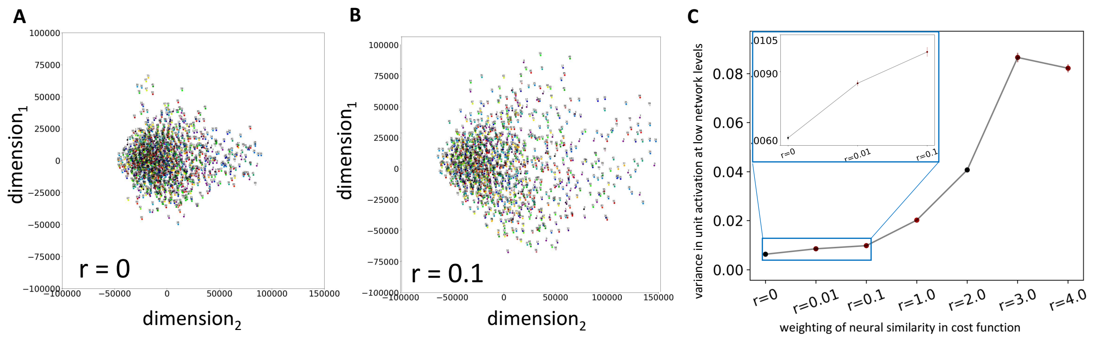


Figure III.4: Visualizing the hidden unit activations. In panels A and B, we input the same 1280 randomly-selected images from the CIFAR100 test set into DCNNs trained with either r=0 or r=0.1. We then used t-SNE to embed those high-dimensional unit activations into two dimensions. X- and Y-axes represent the two dimensions of this t-SNE embedding. Colors indicate the object categories for each input image. A) t-SNE embedding of hidden-unit activations from networks trained with no neural data (r=0). B) t-SNE embedding of hidden-unit activations from networks trained with neural data r=0.1 for the first 10 epochs of training. C) Average variance per unit in the V1-like layer of trained CORNet-Z networks with different weightings, r, of neural representations in the cost function. Inset zooms in on the first data points for $r = 0, 0.01, 0.1$.

III.4.4 The Details Matter

Training neural networks to categorize objects, while matching the image representations seen in monkey V1, leads to improved object recognition performance (Fig. III.2). Is that result specific to the image representations in the monkey brain, or would having *any* arbitrary added RSM constraint in the cost function yield similar results? To answer this question, we performed the same neural network experiment described above (Fig. III.2), but using randomly-generated matrices in place of the monkey V1 representational similarity matrices. For these experiments, we used the optimal cost function weighting ratio, r, found from our experiments with V1 data: r=0.1.

We created randomly-generated RSMs in several different ways (see Methods), used them in place of the monkey V1 RSM in the training procedure, and compared the trained networks' object categorization performance. Networks trained with the Gaussian random-data RSMs (either with or without matching the mean and standard deviation of the monkey data) underperformed ones trained with real V1 data for the teacher RSM. Networks trained with shuffled V1 data get nearly the same testing accuracy as do those trained with real V1 data (Fig. III.5), although the real V1 data still appears to form (by a small margin) the best teacher representation. The teacher RSM may be instructing the DCNN about the distribution of activations (see sec. III.4.3), making the statistics of the data important while not requiring the exact V1 data.

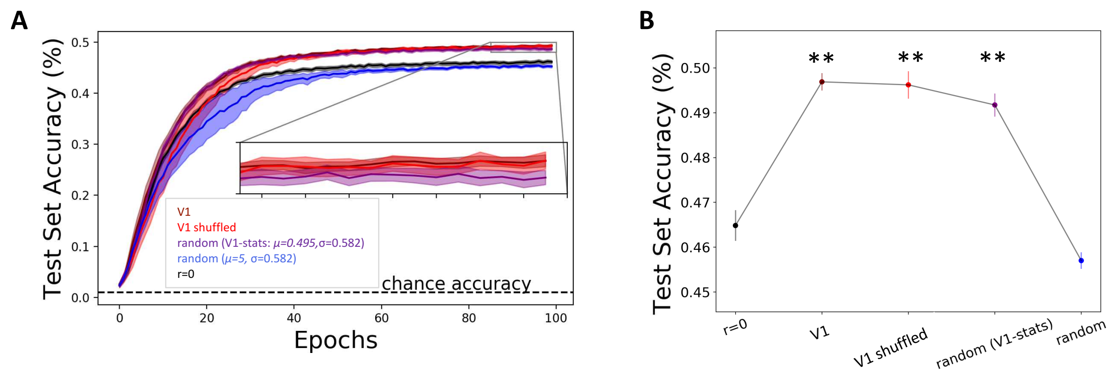


Figure III.5: Accuracy in categorizing previously un-seen CIFAR100 images for networks trained with different teacher RSMs: the real monkey V1 RSM (dark red); the image-shuffled V1 data RSM (light red); RSM from random Gaussian vectors drawn with the same mean and standard deviation as the V1 data (purple); and RSM from random Gaussian vectors drawn with different mean than the neural data (blue). These were all trained with a weighting of $r = 0.1$ applied to the representational similarity in the loss function. For comparison, the baseline network (trained with no representational similarity cost) is shown in black. A) Testing accuracy over epochs of training. Shaded areas on plot are +/- SEM over 10 different random initializations of each model. B) Test accuracy plotted (same as in A) by type of data used in forming the teacher RSM. Double asterisks (**) indicate significantly different results from no neural data, $r = 0$, at p<.001 on a one-tailed t-test. Single asterisks (*) indiciate significantly different results from no neural data, $r = 0$, at p<.05 on a one-tailed t-test.

III.4.5 The Layer Matters

Above, we found that the details of the RSM used as a teacher for training the DCNN, matter. This led us to wonder whether it matters *where* in the network that representational similarity cost is applied. To test this, we repeated the experiments from Fig. III.2 (training CORNet-Z architecture DCNNs with different weightings for the mismatch between network RSM and monkey V1 RSM), but computed the representation similarity cost for layers other than the V1-identified CORNet-Z layer. The resultant networks all had lower training and testing accuracy than did networks trained with no neural data (Fig. III.6). These results, and those in the preceding section, demonstrate that the performance benefits of the monkey V1 representation teacher require that the monkey V1 representation similarity cost be assigned to the appropriate layer in the DCNN.

III.4.6 Reasonable Errors

We demonstrated that teaching neural networks to respond to images in a more brain-like manner boosts accuracy in categorizing held-out testing data (Fig. III.2). All networks, regardless of regularization, still make frequent errors. However, some errors are worse than others. For example, confusing a mouse for a hamster is more reasonable than confusing a mouse for a skyscraper. This intuition led us to quantify the quality of the errors made by each of our trained networks. For this purpose, we exploited

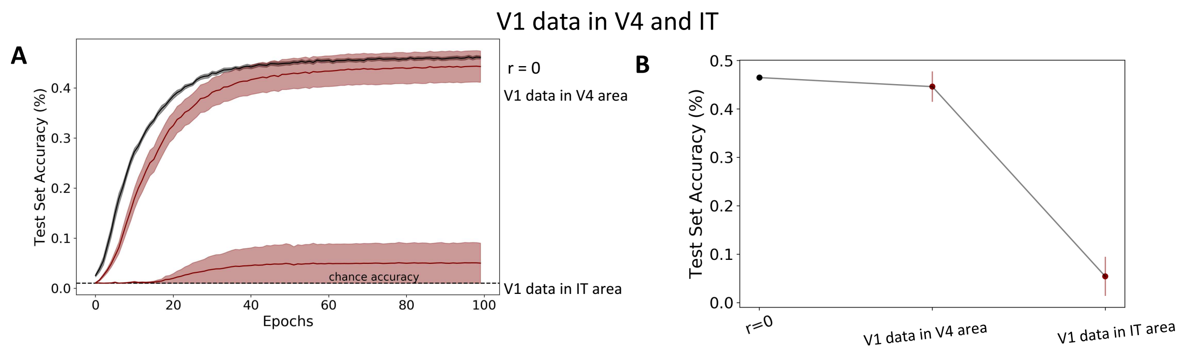


Figure III.6: Accuracy in categorizing previously-unseen CIFAR100 images for the CORNet-Z architecture trained with different weighting ratios, r, applied to monkey V1 representation similarity for all 100 epochs of training in the V4-like and IT-like areas of CORNetZ (see Fig. III.1). A) Testing accuracy over epochs of training. Shaded areas on plot are +/- SEM over 10 different random initializations of each model. B) Test accuracy plotted (same as in A) by type of data used in forming the teacher RSM.

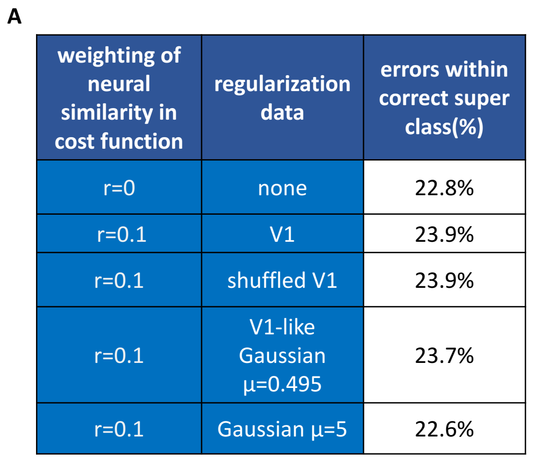
the fact that the 100 classes of labels in the CIFAR100 dataset are grouped into 20 superclasses. One example is “small mammals", which encompasses mouse, rabbit, shrew, and squirrel.

We thus asked, for each trained network, what fraction of their categorization errors were within the correct superclass (e.g., confusing a hamster for a mouse) vs in the wrong superclass (e.g., confusing a mouse for a skyscraper). We performed this test on the network trained with the monkey V1 data as a teacher, with the weighting ratio that yielded the best categorization performance ($r = 0.1$). For this network, errors were within the correct superclass 23.9% of the time (Fig. III.7). For comparison, for the network trained with no neural data, errors were within the correct super class 22.8% of the time (Fig. III.7).

Networks trained to categorize images, using monkey V1 as a teacher make fewer categorization errors (Fig. III.2) and, when they do make errors, those errors are more often within the correct superclass and thus more reasonable (Fig. III.7). We find a similar pattern in networks trained with other weighting ratios r (Fig. III.7 B). Moreover, results with randomized RSMs mirror their impact on categorization: when a teacher RSM improves categorization accuracy, it increases the fraction of errors that are within the correct superclass.

III.4.7 Robustness to Corrupt Training Data

Given that networks trained using neural data made fewer and more reasonable errors, we hypothesized that they generalized more robustly. To test that hypothesis, we did experiments in which some image labels in the training dataset were incorrect (i.e., in the presence of label noise). Datasets will often contain misclassified images, and ideally computer vision networks would not be heavily affected by these


A

weighting of neural similarity in cost function	regularization data	errors within correct super class(%)
r=0	none	22.8%
r=0.1	V1	23.9%
r=0.1	shuffled V1	23.9%
r=0.1	V1-like Gaussian μ=0.495	23.7%
r=0.1	Gaussian μ=5	22.6%

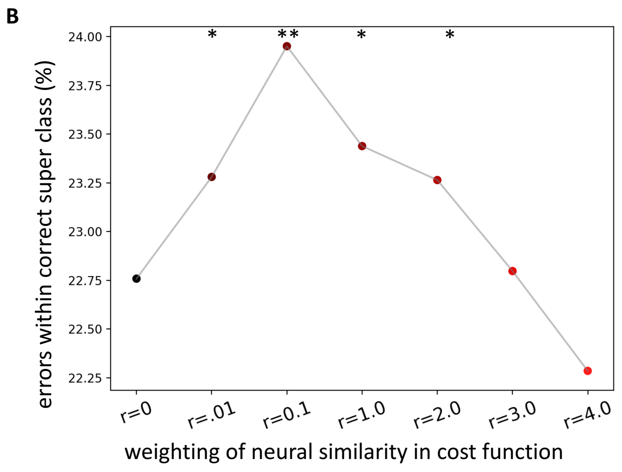


Figure III.7: The percentage of errors that are within the correct superclass in the CIFAR100 dataset. A) Percentage of errors within the correct superclass for the network trained with no neural data (r=0), with the RSM formed by: monkey V1 data; image-shuffled V1 data (as described in sec. III.3.5); data drawn from Gaussian distribution with the same mean and standard deviation as V1 data; and data drawn from Gaussian distribution with a different mean than the neural data. B) Percentage of errors within the correct superclass for networks trained with the monkey V1 RSM as a teacher, and various weightings of neural similarity in the cost function (Eq. III.1) Double asterisks (**) indicate significantly different results from no neural data, $r = 0$, at p<.001 on a one-tailed t-test. Single asterisks (*) indicate significantly different results from no neural data, $r = 0$, at p<.05 on a one-tailed t-test.

misclassifications. To generate a corrupted training set, we switched labels of 10% to 50% of the training dataset, keeping the distribution of classes equal. We only corrupted the training data, and left testing data intact. We then trained CORNet-Z networks with $r = 0.1$, and networks without neural data, r=0. By the end of the 100 training epochs, networks trained with neural data achieved better testing accuracy than did those without neural data (Fig. III.8 B). The generalization error (training cost - testing cost) was also lower for networks trained with neural data (Fig. III.8 C).

III.4.8 Training Over Longer Timescales

In the preceding experiments, we used the neural data regularizer only during the first 10 epochs of the training procedure: after the 10th epoch, the weighting parameter, r, was set to zero. This led us to wonder whether larger performance benefits might be obtained by extending the time window over which the neural data contributed to the training. To answer this question, we trained networks with neural data for all 100 epochs, instead of the first 10 as done in above figures, and found a similar boost in testing accuracy for networks trained with neural data (Figs. III.9 A, B). This demonstrates that we do not need to run the regularization step for longer than the first 10 epochs.

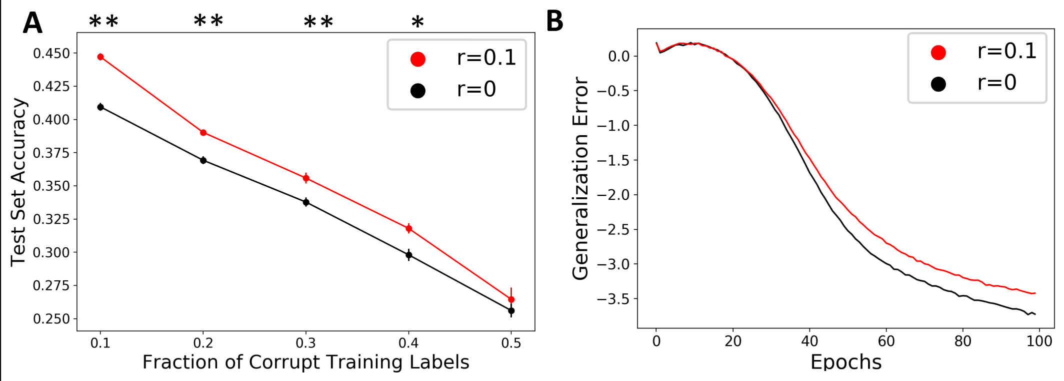


Figure III.8: Training CORNet-Z networks with corrupted labels. A) Test set accuracy on networks trained with (red) and without (black) neural data, with different fractions of corrupted labels in the training set. Labels for the test set were not corrupted. Double asterisks (**) indicate significantly different results from no neural data, $r = 0$, at p<.001 on a one-tailed t-test. Single asterisks (*) indicate significantly different results from no neural data, $r = 0$, at p<.05 on a one-tailed t-test. B) Generalization error (training cost - testing cost) on networks trained with (red) and without (black) neural data and different fractions corrupted training labels. Line patterns indicate different fractions of corrupted labels in the training set.

III.4.9 Training With Various Cost Weightings

We then tried training with neural data for all 100 epochs, but with decaying the emphasis on neural data throughout training (Mcclure and Kriegeskorte (2016)). Initially, the network would emphasize matching neural data with ratios r=0.01, r=0.1, or r=1.0. This amount would decay towards r=0 according to $\lambda_{t+1} = \lambda_t(1 - \frac{t}{t_{max}})$. We found a similar boost in accuracy with training for all 100 epochs with static ratios and dynamic ones (Figs. III.10 C, D).

III.4.10 Training With Static Cost Weightings

To determine if it was required to maintain a certain ratio between the cross entropy and V1 mean squared error in the cost function, we tried training ratios with a static λ value. This means that as the network better learns to match the V1 representations, the cost function will emphasize more on minimizing cross entropy on the images (similar as above, but without a regimented schedule for decay). We found that this method did not work as well as with a decaying λ in (Fig. III.10) or training for the first 10 epochs (Fig. III.2).

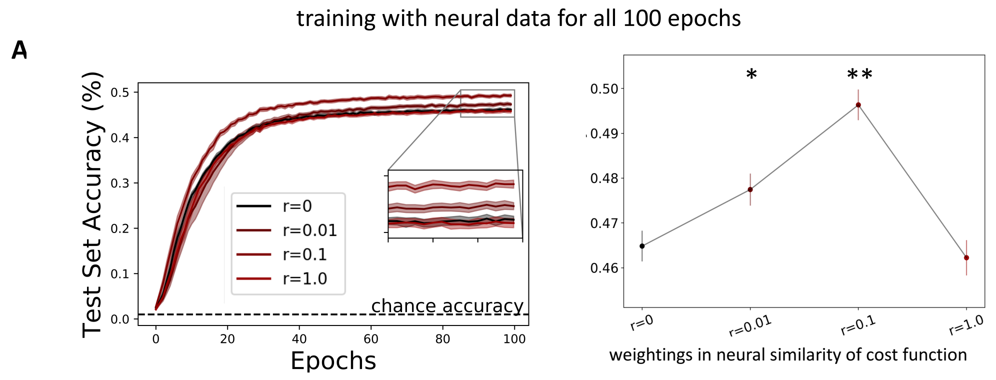


Figure III.9: Training with representation similarity cost at all epochs. (Similar to Fig. III.2 but with representation similarity cost applied for all 100 epochs). A) Testing accuracy for each epoch of training with static weighting ratio, r. B) Test accuracy (same as in A) vs the weight of the emphasis on neural representation similarity. Double asterisks (**) indicate significantly different results from no neural data, $r = 0$, at p<.001 on a one-tailed t-test. Single asterisks (*) indicate significantly different results from no neural data, $r = 0$, at p<.05 on a one-tailed t-test. Chance accuracy is indicated by dashed black lines in A. Shaded areas and vertical bars on plot are +/- SEM over 10 different random initializations of each model.

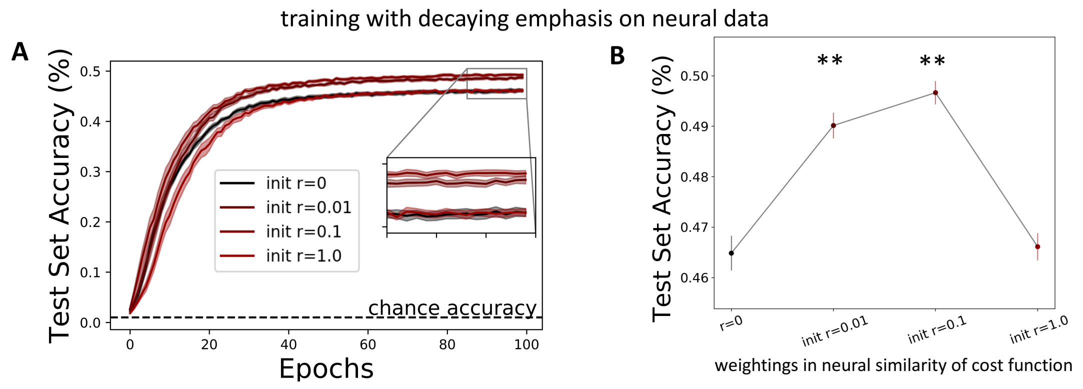


Figure III.10: Training with representation similarity cost at all epochs. (Similar to Fig. III.2 but with representation similarity cost applied for all 100 epochs). A) Testing accuracy for each epoch of training for an initial ratio, init r, which decays during training. B) Testing accuracy (same as in A) vs the weight of the emphasis of neural representation similarity. Double asterisks (**) indicate significantly different results from no neural data, $r = 0$, at p<.001 on a one-tailed t-test. Single asterisks (*) indicate significantly different results from no neural data, $r = 0$, at p<.05 on a one-tailed t-test. Chance accuracy is indicated by dashed black lines in A. Shaded areas and vertical bars on plot are +/- SEM over 10 different random initializations of each model.

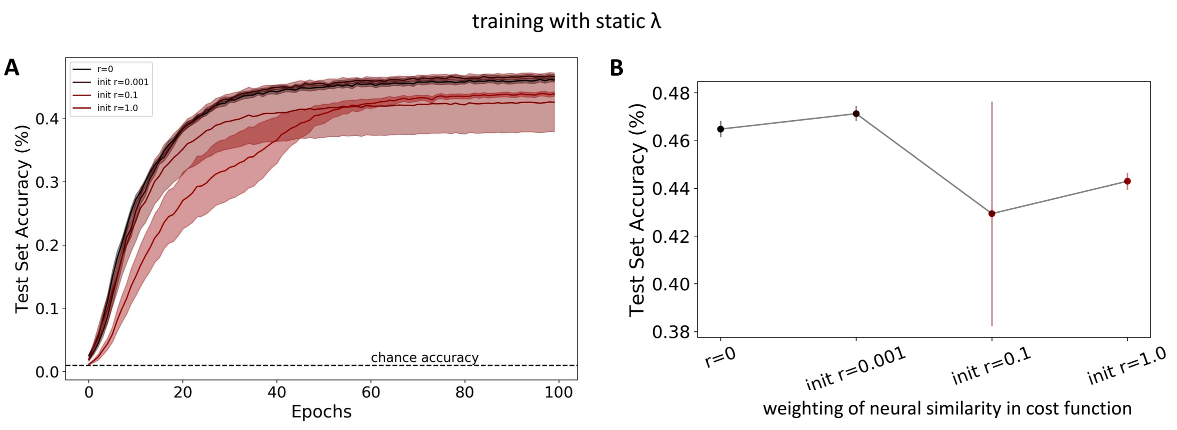


Figure III.11: Accuracy in categorizing previously-unseen CIFAR100 images for the CORNet-Z architecture trained with different initial weighting ratios, *init r*, applied to monkey V1 representation similarity for all 100 epochs of training. The test-set accuracy is plotted at each epoch during training. Chance accuracy indicated by dashed black line. Shaded areas are +/- SEM over 10 different random initializations of each model. B) Test-set accuracy for previously-unseen CIFAR100 images at the end of training, as a function of the initial static weight, *init r*, given to neural representational similarity in the cost function.

III.4.11 Training Accuracy

The accuracy on the set of training images for CIFAR100 with various weightings, r, applied to the monkey V1 representation. Training for no neural data, $r = 0$, and the optimal neural data, $r = 0.01$, is most similar. All weightings converge by the end of training.

III.4.12 Training With V4 and IT

We then wondered if we would be able to get improvements in our object recognition tasks by using teacher signals from deeper visual areas, i.e. V4 and IT. As for V1, we trained neural networks on the composite cost (Eq. III.1), with varying ratios r describing the trade-off between representational similarity cost and categorization cost. We evaluated the trained networks based on categorization accuracy achieved on held-out data (not used in training) from the CIFAR100 dataset. We calculated the RSMs as described in sec. III.3.2. As above, in our figures, black lines indicate networks trained purely for categorization ($r = 0$), while red lines indicate networks trained using monkey V4 or monkey IT as a teacher ($r > 0$). Higher weightings of neural data in the loss function correspond to the shade of red lines. We trained networks with varying parameters and found similar results as displayed in Fig. III.13. We trained these networks with neural data on epochs $50 - 100$. We do not train with neural data in the beginning because we are now applying teacher signals to deeper layers in the DCNN .

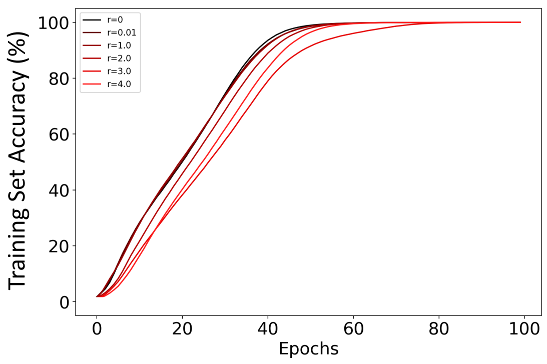


Figure III.12: Accuracy on the training set of images throughout the training procedure on CIFAR100 images for the CORNet-Z architecture trained with different weightins, r, applied to the monkey V1 representation for the first 10 epochs of training. The training-set accuracy is plotted at each epoch during training. Shaded areas are +/- SEM over 10 different random initializations of each model.

We found that networks trained with V4 or IT data resulted in lower or similar testing accuracy than the baseline CORNetZ model. We cannot conclusively state that V4 or IT data would not be helpful in training object recognition networks. We focused our efforts primarily on utilizing V1 as a teacher signal, so the decrease in accuracy may be because we did not find the optimal combination of how many epochs for which to train the network on the joint loss function, and how much to emphasize matching the neural representations. It could also be because the data normalization done within the lab that collected those data (Majaj *et al.* (2015)) was different from the lab from which our V1 data came (Coen-Cagli *et al.* (2015)).

III.4.13 Results on VGG-16 Network

All of the above experiments were obtained on the relatively small CORNet-Z network. This led us to wonder whether monkey V1 data could serve as a similarly effective teacher for deeper neural networks with higher baseline performance. To answer this question, we repeated our above experiments, but using the VGG-16 architecture (Simonyan and Zisserman (2015)) in place of CORNet-Z. We applied the monkey V1 representational similarity cost at the third convolutional layer of the VGG-16 (see Methods). We found that, with the VGG-16 architecture, the monkey V1 teacher signal improves categorization performance, increases robustness to corrupted labels in the training set, and reduces generalization error, similar to what was observed with the smaller CORnet-Z architecture (Fig. III.14).

III.5 Future Work

Important future work includes more rigorous testing in the use of V4 and IT as a teacher signal for training object recognition tasks. We did not have access to V2 data which should also be explored.

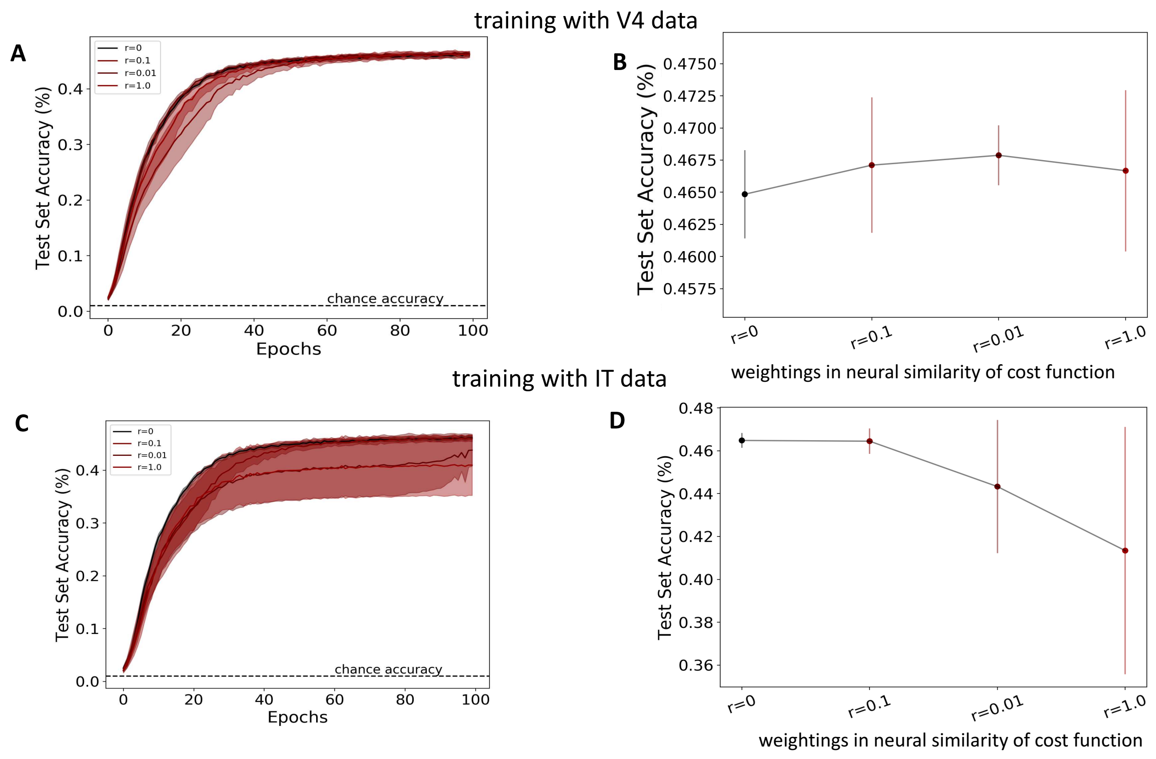


Figure III.13: Accuracy in categorizing previously-unseen CIFAR100 images for the CORNet-Z architecture trained with different weighting ratios, r, applied to monkey V4 (A, B) or IT (C, D) representation similarity. A) Test-set accuracy at each epoch during training with V4 data. Chance accuracy indicated by dashed black line. Shaded areas are +/- SEM over 10 different random initializations of each model. B) Test-set accuracy for previously-unseen CIFAR100 images at the end of training, as a function of the weight (r) given to neural representational similarity in the cost function at the end of training with V4 data. C) Test-set accuracy at each epoch during training with IT data. Chance accuracy indicated by dashed black line. Shaded areas are +/- SEM over 10 different random initializations of each model. D) Test-set accuracy for previously-unseen CIFAR100 images at the end of training with IT data, as a function of the weight (r) given to neural representational similarity in the cost function.

Because our V4 and IT data were normalized in a different way from our V1, we cannot conclude that V4 or IT would not be useful. If future experiments determined V1 was most useful, we hypothesize that this may be because of the more general representations found in V1. It may be more difficult to utilize V4 or IT because we would need recordings from the neurons that respond to the objects in the classification set. Ideally, we would be able to design experiments to record from macaque monkeys on the same dataset as we are classifying on to more rigorously test the idea of using deeper visual layers.

We demonstrated that using monkey V1 as a teacher signal improved performance in both CORNet-Z and VGG-16 network architectures. We did not; however, test the use of V1 teaching signals in larger state-of-the-art achieving networks. Interesting future work could perform these same experiments in deeper CNNs. We also only performed experiments on the 100-class CIFAR100 dataset (Krizhevsky

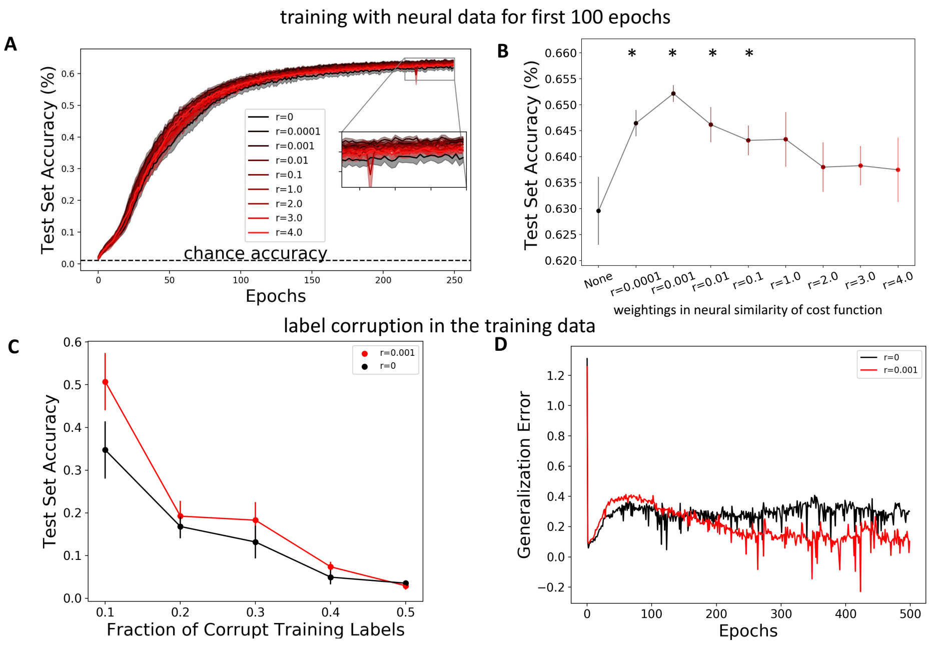


Figure III.14: Accuracy in categorizing previously-unseen CIFAR100 images for the VGG-16 architecture trained on the composite tasks, with different weighting ratios, r, applied to monkey V1 representational similarity in the cost function. A) Testing accuracy for each epoch of training. Chance accuracy indicated by dashed black line. Shaded areas on plot are +/- SEM over 10 different random initializations of each model. B) Test accuracy plotted (same as in A) vs the weight of the emphasis on neural representation similarity. C) Test set accuracy on networks trained with (red) and without (black) neural data and 0.1 to 0.5 fraction corrupted training labels. Labels for the testing set have not been corrupted. D) Generalization error (training cost - testing cost) on networks trained with (red) and without (black) neural data and 0.1 fraction corrupted training labels. In panels B and C, double asterisks (**) indicate significantly different results from no neural data, $r = 0$, at p<.001 on a one-tailed t-test. Single asterisks (*) indicate significantly different results from no neural data, $r = 0$, at p<.05 on a one-tailed t-test.

(2009)). There would be more experiments that could be done if this idea was applied on a network trained on ImageNet (Deng *et al.* (2009a)) with 1000 categories and more room to test the hypothesis that training neural networks with neural data leads to making errors that are more reasonable.

Another important area for future work is to develop a better assessment of which errors are 'reasonable' vs 'unreasonable'. We propose using the built in super-class categories for our experiments in III.4.6, but these are not well studied. To address this, we are currently doing experiments where human subjects perform categorization tasks; those experiments are the subject of ch. 4.

III.6 Discussion

Training the early layers of convolutional neural networks with monkey V1 image representations as a teacher improves those networks' ability to categorize previously-unseen images. Moreover, when networks had the monkey V1 representation teacher, their mistakes were more reasonable than they were for networks with no such teacher representation. These results were consistent for two different network architectures of vastly different sizes (CORNet-Z and VGG-16), suggesting that they will generalize well to other networks. We emphasize that our goal was not to achieve state-of-the-art classification performance, but rather to determine whether and how we could use the brain as a teacher network for object recognition tasks. We tested this with powerful but relatively small networks. Given our positive findings, future work could fruitfully apply this neural data teaching procedure to larger networks to make better object recognition systems.

We also tried training networks with V4 and IT recorded data from macaque monkeys (Majaj *et al.* (2015)). Using the CORNetZ architecture, we experimented with forcing the deeper layers of the network to match representations from those recordings (sec. III.4.12. This resulted in lower testing accuracy than the baseline CORNetZ model. However, we cannot conclusively state that V4 or IT data would not be helpful in training object recognition networks. The decrease in accuracy may be because we did not find the optimal combination of how many epochs for which to train the network on the joint loss function, and how much to emphasize matching the neural representations. It could also be because the data normalization done within the lab that collected those data (Majaj *et al.* (2015)) was different from the lab from which our V1 data came (Coen-Cagli *et al.* (2015)). However, it remains possible that V1 data may be actually more useful for training DCNNs because the specificity of neural responses increases in the deeper layers of the mammalian visual system: the V4 and IT data may not include recordings from neurons that would be useful in our particular object recognition task, whereas V1 neurons carry more generalist representations. Finally, features in DCNNs are only transferable in the early layers and become much less so in deeper layers such as the equivalent to V4 and IT (Yosinski *et al.* (2014)), further suggesting that V1 data could be the most useful brain data for training object recognition networks.

CHAPTER IV
HUMAN CLASSIFICATION ON AN OBJECT RECOGNITION DATASET

IV.1 Summary

Object recognition tasks are missing a key metric of evaluation: how human-like are the errors made by the computer vision algorithms? We present a dataset that can be used to evaluate the reasonable-ness of errors made by these algorithms. We asked individuals, paid via Mechanical Turk, to rank how similar objects in images in the ImageNet 1000-class dataset from 1-9 (1 being completely different and 9 being highly similar). This can be used to better compare computer vision algorithms as well as for training computer vision algorithms by weighting errors that are less reasonable higher than other errors.

IV.2 Introduction

Object recognition is the most basic aspect of computer vision problems, yet there is still significant room for improvement on both the side of the algorithms and the task set-up itself. Here, we focus on how to improve the task setup by incorporating an additional metric for evaluating our algorithms: how human-like were the mistakes made by the network? To create this new metric, we will start with the largest object recognition dataset, ImageNet (Deng *et al.* (2009b)), used for comparing state-of-the-art computer vision algorithms, and ask humans to determine which classes are most similar.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Deng *et al.* (2009b)) contains over 1 million images separated into 1000 classes. These classes vary from animal species including cranes, sea slugs, and German shepherd dogs to household items such as measuring cups and necklaces to outdoor scenes such as seashores and mountain ranges. The ImageNet dataset was collected by querying the keyword of the classification name and then having humans manually categorize the image with a binary 'yes/no', for whether or not the object appears in the image. This creates a lot of room for errors and discrepancies in the dataset. Humans sifting through over 1 million images are likely to make many mistakes and incorrectly deem an image as containing the object. Another issue is that these images were selected based on a binary classification, whereas the actual task is a 1000-classification problem. This creates problems where an image of a person eating a lobster can get classified as 'lobster' because it does in fact contain a lobster, but from an algorithmic perspective, there is not clear information on which object in the image (lobster vs. human) is more important.

Beyond issues with the dataset, there remains problems with how we evaluate the accuracy of our computer vision algorithms. Currently, ImageNet state-of-the-art results are demonstrated based on either top-1 or top-5 accuracy. Computer vision algorithms output a probability where, for each image,

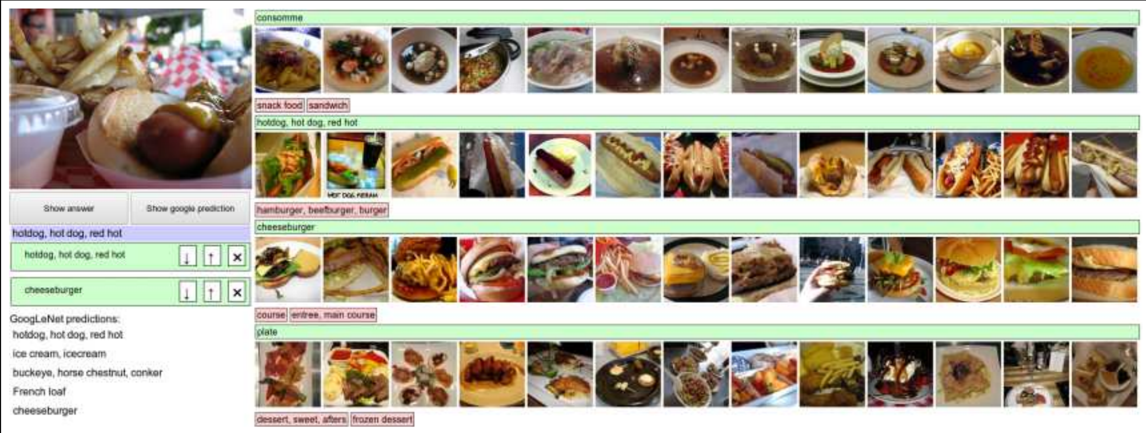
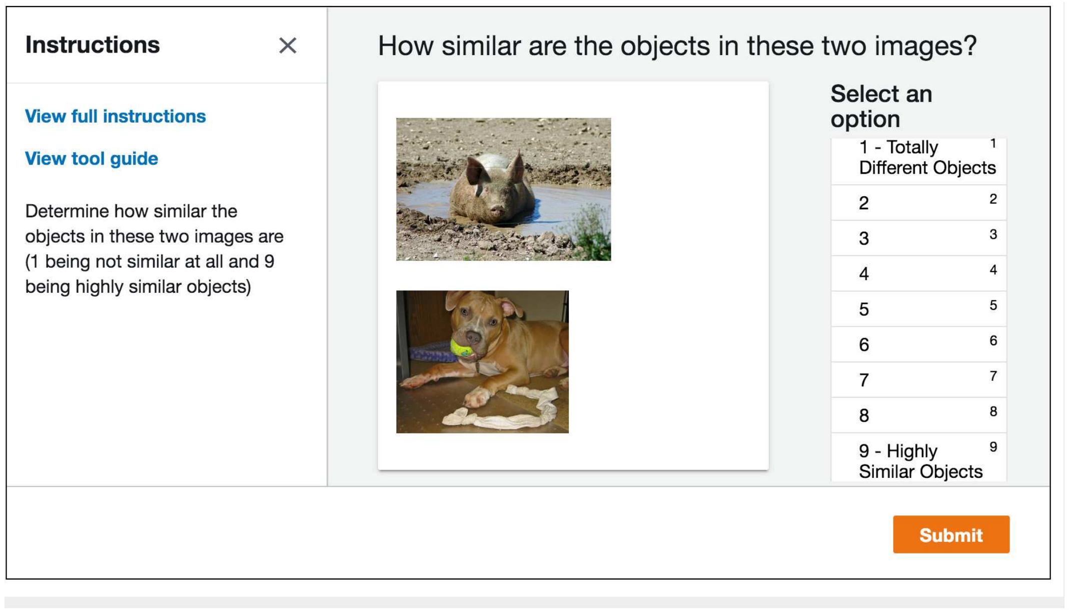


Figure IV.1: Previous work from Andrej Karpathy on gathering human classification data on the ImageNet dataset. The test image appears on the left and all 1000 classes appear on the right with 13 sample images.

each of the 1000 classes is given a probability score of how likely that image is to belong to that class. Top-1 accuracy indicates that the number one class output by the algorithm is the proposed class, so a correct output gives 100% accuracy whereas any other class gives 0%. This is a problem for multiple reasons. Suppose for example the image in question was a golden retriever and the algorithm output 'Labrador retriever' as the most likely class. We are calling this a completely wrong answer. Suppose another computer vision algorithm labeled the 'golden retriever' as a 'mountain range'. Our current metric of evaluation would consider these two outputs equally wrong.

The top-5 accuracy was proposed as a way to be more flexible in evaluating the outputs of computer vision algorithms. In top-5 accuracy, we look at the top 5 most likely classes output by the algorithm. If any of these top-5 likely classes is the correct class, we call that 100% accuracy. This presents a series of its own challenges. Suppose the algorithm that labeled the 'golden retriever' as 'mountain range' had a top 5 as follows: 1. mountain range 2. necklace 3. ant 4. scuba diver 5. golden retriever. Suppose another algorithm output the top-5 most likely classes as: 1. Labrador retriever 2. golden retriever 3. flat coated retriever 4. German Shepherd 5. English Springer. The top-5 accuracy will reflect these networks as having identical performance while it seems fairly obvious that the algorithm that output all breeds of dogs as the top-5 should be regarded as having a better performance.

Previous work on getting human classification on the ImageNet dataset found 5.1% top-5 errors on a subset of the data (Karpathy (2014)) from one individual. This was collected by displaying one test image and all 1000 class options on the right hand side (Fig. IV.1). The individual classifying the images trained on 500 images and then tested himself on 1500 images. This is our best record of human classification to date, but still suffers from lack of in depth coverage in the dataset, only being performed


Instructions
View full instructions
View tool guide
How similar are the objects in these two images?
Select an option
1 - Totally Different Objects
2
3
4
5
6
7
8
9 - Highly Similar Objects
Submit

Determine how similar the objects in these two images are (1 being not similar at all and 9 being highly similar objects)

Figure IV.2: A sample of one 'hit' an mturker will complete. In this example, we are asking the mturker to compare classes n02395406 ('hog') and n02093428 ('American Staffordshire terrier').

by one person, and requiring knowledge of the names of objects to know to search for them. For example, there are 250 dog breeds in the dataset, so even if he knew two dogs were different breeds when compared to one another, he may not know which dog is which offhand. The state-of-the-art object recognition in computer vision algorithms is 2.9% top-5 errors (Tan and Le (2019)).

Some errors seem clearly more reasonable from a human-perspective, but we need an objective way of evaluating this. Here we propose evaluating the reasonableness of errors by determining how similar the mistaken class is to the correct class. We do this by asking people to determine how similar two images from different classes are. We present people (via Mechanical Turk – we'll call mturkers) with two images from different classes within ImageNet and a scale from 1-9. We ask them to assign a value to how similar the two objects in the images are, 1 meaning the objects are totally different and 9 meaning highly similar. We stop at 9 because we never present two images from the same class so no objects should be evaluated as the exact same (or a 10). We can use these responses to determine if two classes are reasonable mistakes for each other or not.

IV.3 Methods

In order to get an in-depth and reasonably objective understanding of what human-like mistakes would be for the ImageNet dataset, we asked people to rank how similar two images were from different object classes within the dataset (Fig. IV.2). To choose the sample images for each class, we randomly downloaded 5 images from the ImageNet database and then manually selected the most representative

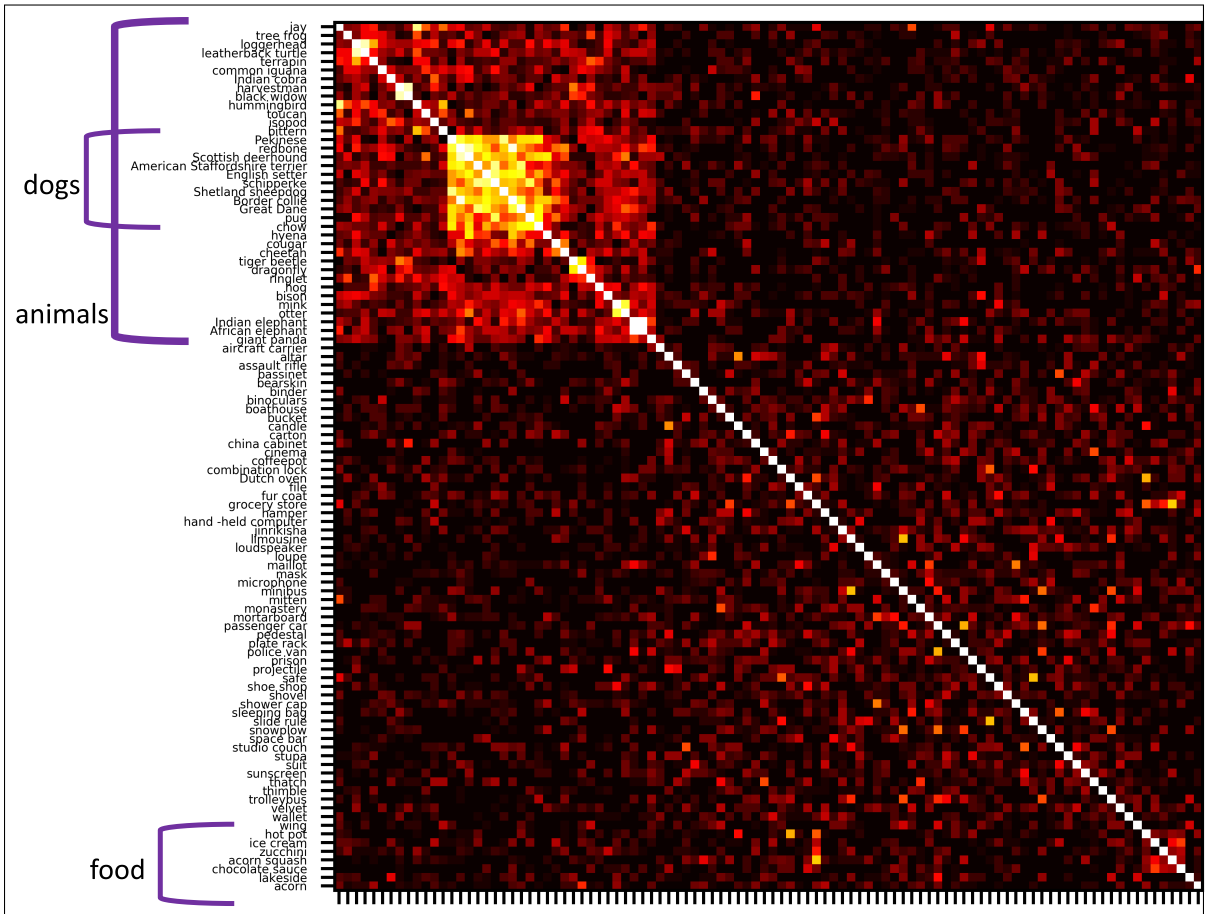


Figure IV.3: Classification responses. Heatmap of the similarity scores evaluated by the mturkers where lighter colors indicate more similar objects. The heatmap was sorted based on scores prior to plotting. The heatmap is displayed as symmetrical responses although only one half of the heatmap was directly inquired. The diagonals are filled in as identical. The class labels are listed on the left with some clusters indicated ('animals', 'dogs' and 'food').

image. Criteria for selecting the most representative image was based on ensuring the object that was being classified was the main point of the image (either that there were no other objects that may be confused for classes in the image or that the main object was the largest), the main object was large enough in the photo, and the image itself was high enough resolution.

For proof of concept, we randomly selected 100 classes from the dataset and asked mturkers to compare one image from each class with every other class. I requested 5 unique mturkers to assess each comparison of images to see how varying the responses were and get a better representation of responses. We created all possible combinations of hits, but do not perform any experiments on evaluating within

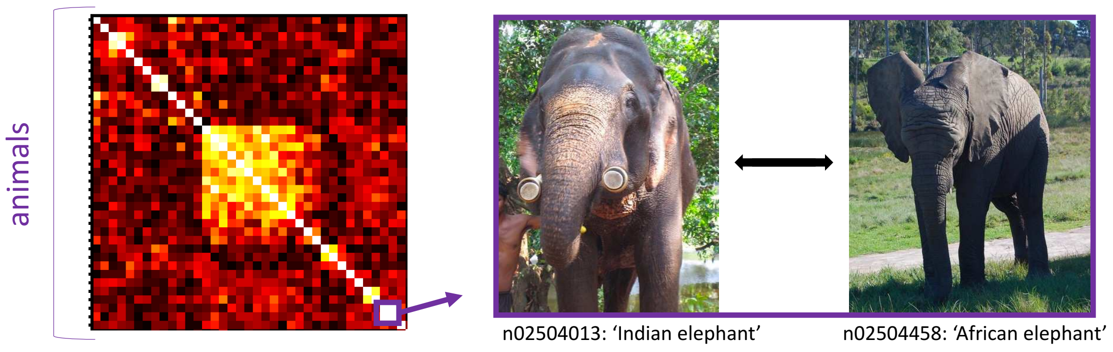


Figure IV.4: Zoomed in on the cluster of animal classes in the heatmap from Fig. IV.3. We then show an example of two classes unanimously evaluated as being the exact same object.

class similarity (i.e. presenting the worker with two images from the same class) or if order affects answers (i.e. if we show class1 before class2, would we get the same score as presenting class2 before class1).

IV.4 Results

We evaluated the responses for all pairs of classes and visualized it in a heatmap (Fig. IV.3). The entry in row i and column j of the matrix is the similarity score between images of objects in categories i and j. These scores were summed over the 5 unique mturker responses. We filled in the symmetrical values for the matrix which were not explicitly questioned, in other words we are ignoring order of classes for similarity and assuming they would be the same regardless of order. We also filled in the diagonal of the matrix as identical images. Lighter colors in the heatmap indicate classes that are more similar.

The clusters that emerge from the heatmap give us an indication we are getting some reasonable signal from our responses. To further show this, we found one pair of images or classes that were ranked as identical by all mturkers - 'n02504013' or 'Indian elephant' and 'n02504458' or 'African elephant' (Fig. IV.4). We also looked at the histogram of responses to ensure all values (1-9) were used and see that they are IV.5. Most objects are rated as '1 - Totally different objects'. We then calculated how much variance there was from the 5 unique mturkers responses and found $\sigma^2 = 0.703$, suggesting we are getting reasonably similar responses from the mturkers. Further work needs to be done to evaluate variance.

IV.5 Future Work

Important future work includes collecting the full 1000-category comparison dataset. Before moving to the full dataset; however, we considered how the answers would depend on the way the question was posed. We started with small experiments of 20 classes and changed the verbiage and/or scale to evaluate which method should be used as we scale up to the full dataset. In order to compare and select the most useful survey style, we use the Kullback-Leibler divergence (or K-L divergence, also known as relative

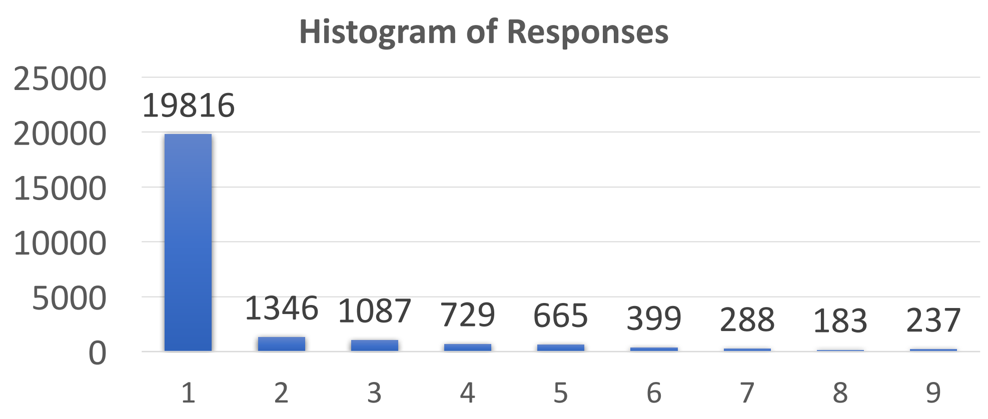
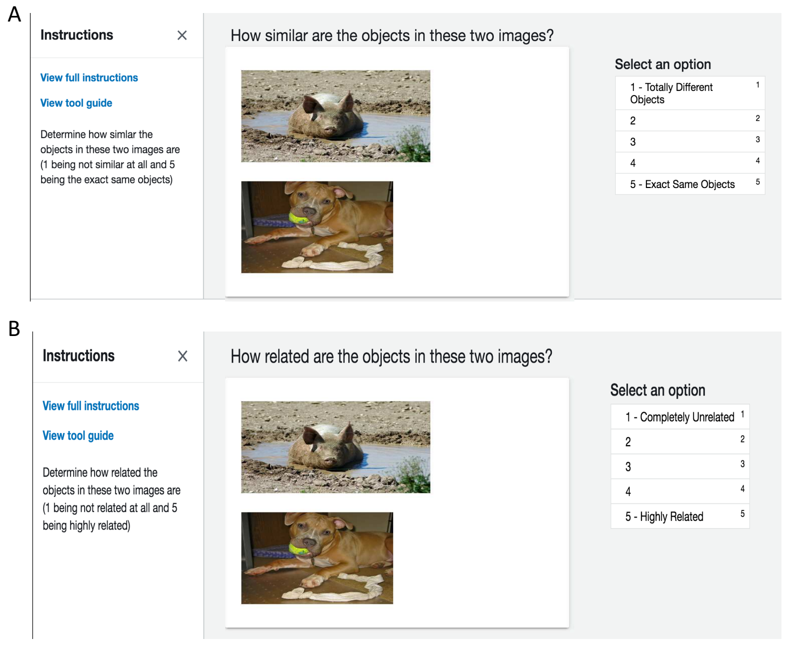


Figure IV.5: Histogram of individual responses. Response options were 1 (totally different objects) to 9 (highly similar objects). The total number of responses was 24,750.

entropy) (Kullback and Leibler (1951)). The K-L divergence allows us to determine how similar the responses of the various experiments were to a distribution. In this case, we wanted the responses of the questionnaire to be closest to the uniform distribution, meaning we were getting as wide a range of responses for similarity as possible.

We first considered whether a 1-5 scale would give a more uniform-distribution set of responses (Fig. IV.6 A). We cut the 100-class experiment using the 1-9 scale down to the equivalent 20-class experiment and found the K-L divergence score compared to the uniform distribution was -1778.79. For the experiment using the 1-5 scale (sample question depicted in Fig. IV.6 A), the K-L divergence score was -530.09, indicating it is closer to the uniform distribution. This means the range of answers from 1-5 get used more often making the dataset likely to be more useful. We also considered how important the verbiage 'similar' was in our question – 'How *similar* are the objects in these two images?'. We repeated the same 20-class experiment using *related* instead – 'How *related* are the objects in these two images?' (Fig. IV.6 B). We found that the K-L divergence score for this experiment was -482.51, indiciating it is slightly closer to the uniform distribution than when we used the term 'similar'. Important future work will be to determine if other variations of the question should be considered, and what the best metric is to decide which one to use for the full experiment. One example would be to ask instead, 'How surprised would you be if the objects in these two images were mistaken for each other?'. This might get us closer to the goal of the dataset, asking how *bad* of an error it would be for a self-driving car to mistake two objects.

Based on our current work using the 100-class experiment, we can estimate the number of individuals required to scale each pair of classes based on how much variance we deem acceptable. For example, if we take the variance between two similar dog classes and compare this to the variance between a dog



Instructions

View full instructions

View tool guide

Determine how simlar the objects in these two images are (1 being not similar at all and 5 being the exact same objects)

How similar are the objects in these two images?

Select an option

1 - Totally Different Objects	1
2	2
3	3
4	4
5 - Exact Same Objects	5

Instructions

View full instructions

View tool guide

Determine how related the objects in these two images are (1 being not related at all and 5 being highly related)

How related are the objects in these two images?

Select an option

1 - Completely Unrelated	1
2	2
3	3
4	4
5 - Highly Related	5

Figure IV.6

class and the next most similar not-dog class, this could give us a reasonable estimate for variance that is needed to distinguish between classes. From this, we can calculate how many individuals are needed for the full dataset. Future work should also include within class scaling (having individuals scale two different images from the same class) to further measure consistency and variance.

This new dataset can be used as a metric to evaluate how reasonable errors in the state-of-the-art computer vision algorithms are. Instead of weighting all errors as equal, important future work would instead weight the severity of errors based on some multiplier based on how similar humans find these classes to be. For example, when 'African elephant' is mistaken for 'Indian elephant', this should be considered a much smaller error than if 'Indian elephant' is mistaken for 'American Staffordshire terrier'.

We can also use this dataset and compare word2vec embeddings. Word2vec embeddings give us a vector space of each word with more similar words being more closer in vector space. We could compare similarities of these embeddings to the similarity of images as ranked by users.

Beyond evaluating the reasonableness of errors, this new dataset could be used to hierarchically train computer vision algorithms. Some previous work has been done on this using the CIFAR100 coarse and fine classes (Zhu and Bain (2017)), but this work suffers from the same limitations as using the coarse labels from CIFAR100 as 'reasonable errors' discussed in chapter 3 – these have not been evaluated by humans. By training a network on a hierarchy of classes that has been evaluated by humans, we could reduce overall error and/or make more reasonable and less catastrophic errors. The idea would be to assign an error value based on how different (or unrelated) humans ranked two classes. For example, mistaking a cat for a dog would likely give a smaller error term than mistaking a cat for a rocket ship.

IV.6 Discussion

We present here a new metric for evaluating and potentially training computer vision algorithms by incorporating the human-decided similarity of classes. This work provides us with a better understanding of how to characterize errors made by computer vision algorithms and a better way to compare and evaluate state-of-the-art algorithms.

There remains important future work as well as some caveats. There is inherent bias in the selection of images to represent each class as one individual (me) selected what was considered to be a representative image out of 5 possible images. There are also potential biases to be considered on the side of the mturkers evaluating the similarity of images. While instructions stated to ignore the background information and focus on what was, in their opinion, the main object in the image, people may ignore these instructions or have various ideas on what the main object is. There are also issues with incentivizing users to spend much time on each hit. The only cut-off used to reject turker responses was if they only selected one number from 1-9 for each hit. Because there is no ground truth it is difficult to evaluate whether or not mturkers spent adequate time looking at the images. With these caveats in mind, it's worth noting that we do find good signal from our responses based on the clusters found in the heatmap (Fig. IV.3), the relatively small variance, and the identical responses from all mturkers on two elephant classes (Fig. IV.4).

CHAPTER V
DISCUSSION

The goal of this research was to contribute towards bridging the fields of artificial intelligence (AI) and neuroscience, or in other words, to put more *neural* in artificial neural networks (ANNs). This allows us to use ANNs to model and better understand the brain as well as using the brain as inspiration for new algorithms and training techniques in AI. We demonstrated the mutual benefit of studying both fields in two problems: storing information for working memory and object recognition in computer vision.

V.1 Storing Information for Working Memory

Working memory requires information about external stimuli to be represented in the brain even after those stimuli go away. In chapter 2, we derived biologically plausible synaptic plasticity rules through which recurrent neural networks self-organize to store information for working memory. We found that networks implementing these plasticity rules were robust to various sources of noise and could store information about multiple stimuli. This provides us with a better understanding of how we store information for working memory which can lead to better insight for understanding deficits in working memory and related neurological diseases and learning disorders. This work is also relevant in machine learning. Machine learning algorithms are often unrealistic from a biological perspective: most rely on training techniques that are not plausibly implemented in the brain. We show that recurrent networks can learn to store information using biologically plausible learning rules. Important caveats to address in this work will include further evaluation on the robustness in the more realistic spike-based model as well as scaling up to a more difficult task than storing information for working memory. This work can be scaled up for more difficult tasks and utilized to train neural networks in more biological and efficient ways. Efficiency is a key area of growth needed for machine learning, algorithms are typically quite slow to train.

V.2 Object Recognition in Computer Vision

Object recognition algorithms are key for applications such as self-driving cars, and still have a long way to go. The state-of-the-art computer vision algorithms, deep convolutional neural networks (DCNNs), are inspired by the architecture of the mammalian visual system (Hubel and Wiesel (1968)) and as they improve at object recognition tasks, they develop representations in their hidden layers that become more similar to those observed in mammalian brains (Yamins *et al.* (2014)). In chapter 3 we found that training computer vision algorithms with monkey V1 neural representations as a teacher signal improves those networks' accuracy on object recognition tasks, leads to more reasonable errors and makes networks more robust to corrupt training data.

We also found this was consistent for two different network architectures of varying sizes, suggesting this new way of training computer vision algorithms will be generally useful. Our results outline a new way to regularize object recognition networks, using transfer learning strategies in which the brain serves as a teacher for training DCNNs. This work can also be utilized in neuroscience to work towards a better model of mammalian vision. Our work was limited due to the current available dataset and computational resources. Important future work will include collecting and utilizing a dataset featuring a more targeted task, for example awake monkeys looking at the same images the CNNs are classifying, as well as implementing the same set-up in state-of-the-art networks with sufficient hyperparametization to determine the best uses for neural data in training neural networks.

In chapter 4, we provide a new metric for object recognition problems that incorporates reasonableness (or human-likeness) of errors. This will allow us to continue to compare computer vision to mammalian vision both for practical applications like the safety of self-driving cars and for studying the visual system. Caveats to this work include biases introduced by the choice of images and verbiage used for survey responses on image classification. Important future work will determine the consequences of various word choices and what the ideal questionnaire should look like.

V.3 Final Remarks

There remains critical work in bridging the fields of neuroscience and AI. More work needs to be done in computational neuroscience to convince the neuroscientists of the value of using machine learning algorithms as models, as well as by the machine learning field to appreciate the value of using the brain as inspiration for algorithms and training techniques.

We present in this work two different strategies towards more brain-like AI. In chapter 2, we focus on training neural networks with learning rules that are biologically plausible and could be implemented in the brain. In chapter 3, we instead focus on making the AI algorithm representations, or outputs, to be more brain-like, regardless of the biologically plausibility of the learning rules. Chapter 4 focuses on better evaluations for the brain-like-ness of our AI algorithms.

This push of making machine learning algorithms more brain-like could have dramatic results in both the field of AI and neuroscience. In AI, we are still not achieving human-level performance on many tasks, and even when we are, we are often not able to do multiple tasks with the same neural network. The more brain-like we push our algorithms to be, the closer to human-level intelligence and performance we are likely to be able to get. The benefits of better AI algorithms are countless. We mention here self-driving cars, but this is just one exciting area where we can autonomize and minimize human error. AI can also revolutionize health care, with one clear example being in elder care. Elder care can be greatly improved with at home health monitoring to better detect health issues without needing to be in a hospital, assisted daily living robots such as fall detection and virtual companions to keep people active and engaged.

In the field of neuroscience, more brain-like machine learning algorithms will allow us to test many new hypotheses quicker and more efficiently. The more brain-like our neural networks, the more likely they are to be able to respond to tests and treatments like an actual brain would, saving us significant amounts of money, lab space and time. Brain-computer interfaces (BCIs) have already helped restore some capabilities lost in neurological diseases such as the ability to speak, move and see by combining AI learning with understanding how our neurons encode sight, sound and movement cues. These are just a small subset of the ways AI can transform our world for the better when we keep neuroscience in the conversation.

References

(1958). "NEW NAVY DEVICE LEARNS BY DOING; Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser." *The New York Times.*

Alberoni M, Baddeley A, Della S, R L, Spinnler H (1992). "Keeping track of a conversation: Impairments in Alzheimer's Disease." *Int. J. Geriatr. Psychiatry.*

Alemi A, Baldassi C, Brunel N, Zecchina R (2015). "A Three-Threshold Learning Rule Approaches the Maximal Capacity of Recurrent Neural Networks." *PLOS Computat. Biol.*

Atkinson R, Shiffrin R (1968). "Human Memory: A Proposed System and its Control Processes." *J Exp Psychol Learn Mem Cogn.*

Baddeley A, Hitch G (1974). "Working Memory." *J Exp Psychol Learn Mem Cogn.*

Bartunov S, Santoro A, Richards B, Hinton G, Lillicrap T (2018). "Assessing the Scalability of Biologically-Motivated Deep Learning Algorithms and Architectures." *ICML.*

Bourdoukan R, Barrett DGT, Machens CK, Deneve S (2012). "Learning optimal spike-based representations." *Adv. Neural Inf. Process. Syst.*

Bourdoukan R, Deneve S (2015). "Enforcing balance allows local supervised learning in spiking recurrent networks." *Adv. Neural Inf. Process. Syst.*

Brendel W, Bourdoukan R, Vertechi P, Machens CK, Deneve S (2017). "Learning to represent signals spike by spike." *arXiv.*

Brody C, Hernández A, Zainos A, Romo R (2003a). "Timing and neural encoding of somatosensory parametric working memory in macaque prefrontal cortex." *Cereb Cortex.*

Brody C, Romo R, Kepecs A (2003b). "Basic mechanisms for graded persistent activity: Discrete attractors, continuous attractors, and dynamic representations." *Curr Opin Neurobiol.*

Cadena S, Denfield G, Walker E, Gatys L, Tolias A, Bethge M, Ecker A (2017). "Deep convolutional models improve predictions of macaque V1 responses to natural images ." *bioRxiv.*

Carandini M (2004). "Amplification of trial-to-trial response variability by neurons in visual cortex." *PLoS Biology.*

Chumbley J, Dolan R, Friston K (2007). "Attractor models of working memory and their modulation by reward." *Biol Cybern.*

Clowes M (1971). "On seeing things." *Artificial Intelligence.*

Coen-Cagli R, Kohn A, Schwartz O (2015). "Flexible gating of contextual influences in natural vision." *Nat. Neuro.*

Constantinidis C, Klingberg T (2016). "The neuroscience of working memory capacity and training." *Nat Rev Neurosci.*

Cowan N (2010). "The Magical Mystery Four: How is Working Memory Capacity Limited, and Why?" *Curr Dir Psychol Sci.*

Crick F (1989). "The recent excitement about neural networks." *Nature.*

Dambre J, Verstraeten D, Schrauwen B, Massar S (2012). "Information Processing Capacity of Dynamical Systems." *Scientific Reports.*

Deneve S, Alemi A, Bourdoukan R (2017). "The Brain as an Efficient and Robust Adaptive Learner." *Neuron.*

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009a). "ImageNet: A Large-Scale Hierarchical Image Database." In *CVPR09.*

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009b). "ImageNet: A Large-Scale Hierarchical Image Database." In *CVPR09.*

Diamond A (2012). “Executive Functions.” *Annu. Rev. Psych.*

Druckmann S, Chklovskii D (2012). “Neuronal circuits underlying persistent representations despite time varying activity.” *Curr Biol.*

Engelhardt P, Nigg J, Carr L, Ferreira F (2017). “Cognitive Inhibition and Working Memory in Attention-Deficit/Hyperactivity Disorder.” *J. Abnorm. Pyschol.*

Federer C, Xu X, Fyshe A, Zylberberg J (2019). “Training neural networks to have brain-like representations improves object recognition performance.” *arXiv.*

Federer C, Zylberberg J (2018). “A self-organizing short-term dynamical memory network.” *Neural Networks.*

Fukushima K (1980). “Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position.” *Biological Cybernetics.*

Funahashi S, Chafee M, Goldman-Rakic P (1993). “Prefrontal neuronal activity in rhesus monkeys performing a delayed anti-saccade task.” *Nature.*

Fuster JM, Alexander GE (1971). “Neuron Activity Related to Short-Term Memory.” *Science.*

Geirhos R, Janssen D, Sch H, Rauber J, Bethge M, Wichmann F (2018). “Comparing deep neural networks against humans : object recognition when the signal gets weaker.” *arXiv.* arXiv:1706.06969v1.

Geirhos R, Michaelis C, Rubisch P (2019). “ImageNet-Trained CNNs are Biased Towards Texture; Increasing Shape Bias Improves Accuracy and Robustness.” *ICLR.*

Goodfellow IJ, Shlens J, Szegedy C (2015). “Explaining and Harnessing Adversarial Examples.” *ICLR.*

Grossberg S (1987). “Competitive Learning: From Interactive Activation to Adaptive Resonance.” *Cognitive Science.*

Gu U, van Gerven M (2015). “Deep Neural Networks Reveal a Gradient in the Complexity.” *J. Neurosci.*

Guergiuev J, Lillicrap T, Richards B (2017). “Towards deep learning with segregated dendrites.” *eLife.*

Hassabis D, Kumaran D, Summerfield C, Botvinick M (2017). “Neuroscience-Inspired Artificial Intelligence.” *Neuron Review.*

Hatfield K (2010). “Binding in visual working memory: The role of the episodic buffer.” *Neuropsychologia.*

He K, Huertas M, Hong S, Tie X, Hell J, Shouval H, Kirkwood A (2015). “Distinct Eligibility Traces for LTP and LTD in Cortical Synapses.” *Neuron.*

Hebb D (1949). *The Organization of Behavior.* New York: Wiley and Sons.

Hellwig B (2000). “A quantative analysis of the local connectivity between pyramidal neuron in layers 2/3 of the visual cortex.” *Biol. Cybern.*

Hinton G, Vinyals O, Dean J (2015). “Distilling the Knowledge in a Neural Network.” *arXiv.*

Hubel D, Wiesel T (1968). “Receptive Fields and Functional Architecture of Monkey Striate Cortex.” *J Physiol.*

Huffman D (1971). “Impossible objects as nonsense sentences.” *Machine Intelligence.*

Huntley J, Howard R (2010). “Working memory in early Alzheimer’s disease: a neuropsychological review.” *Am. J. Geriatr. Psychiatry.*

Inubushi M, Yoshimura K (2017). “Reservoir Computing Beyond Memory-Nonlinearity Trade-off.” *Scientific Reports.*

Karpathy A (2014). “What I learned from competing against a ConvNet on ImageNet.”

Kindel W, Christensen E, Zylberberg J (2019). “Using deep learning to probe the neural code for images in primary visual cortex.” *J. Vis.*

Knag P, Kim J, Chen T, Zhang Z (2015). “A Sparse Coding Neural Network ASIC With On-Chip Learning for Feature Extraction and Encoding.” *IEEE J. Solid-State Circuits*.

Kriegeskorte N (2014). “Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation.” *PLOS*.

Kriegeskorte N, Kievit R (2013). “Representational geometry: integrating cognition, computation, and the brain.” *Trends Cogn Sci*.

Kriegeskorte N, Mur M, Bandettini P (2008). “Representational similarity analysis – connecting the branches of systems neuroscience.” *Front Syst Neurosci*.

Krizhevsky A (2009). “Learning Multiple Layers of Features from Tiny Images.” *Technical report*.

Krizhevsky A, Hinton G (2015). “ImageNet Classification with Deep Convolutional Neural Networks.” *NIPS*.

Kubilius J, Schrimpf M, Nayebi A, Bear D, Yamins D, Dicarlo J (2018). “CORnet : Modeling the Neural Mechanisms of Core Object Recognition.” *bioRxiv*.

Kullback S, Leibler R (1951). “On information and sufficiency.” *Annals of Mathematical Statistics*.

Lecun Y, Bengio Y, Hinton G (2015). “Deep Learning.” *Nature*.

Ledoux E, Brunel N (2011). “Dynamics of networks of excitatory and inhibitory neurons in response to time-dependent inputs.” *Front Comput Neurosci*.

Li N, Daie K, Svoboda K, Druckmann S (2016). “Robust neuronal dynamics in premotor cortex during motor planning.” *Nature*.

Lillicrap T, Cownden D, Tweed D, Akerman C (2016). “Random feedback weights support learning in deep neural networks.” *Nat. Commun*.

Ma W, Husain M, Bays P (2014). “Changing concepts of working memory.” *Nat. Neuro*.

Mackey MC, Glass L (1977). “Oscillation and chaos in physiological control systems.”

Majaj N, Hong H, Solomon E, Dicarlo J (2015). “Simple Learned Weighted Sums of Inferior Temporal Neuronal Firing Rates Accurately Predict Human Core Object Recognition Performance.” *J. Neurosci*.

Marzen S (2017). “Difference between memory and prediction in linear recurrent networks.” *Phys. Rev. E*.

Mcclure P, Kriegeskorte N (2016). “Representational Distance Learning for Deep Neural Networks.” *Front. in Comp. Neuro*.

McCulloch W, Pitts W (1943). “A logical calculus of the ideas immanent in nervous activity.” *Bulletin of Mathematical Biophysics*.

Meunier C, Chameau P, Fossier PM (2017). “Modulation of Synaptic Plasticity in the Cortex Needs to Understand All the Players.” *Front. Synaptic Neurosci*.

Miller E, Fusi S (2013). “Limber neurons for a nimble mind.” *Neuron*.

Morris T (2004). *Computer Vision and Image Processing*. Palgrave Macmillan.

Murphy J, Fuat A, Littlewood E, Conway B, Gathercole S (2010). “Working memory deficits in treated and untreated hypertension.” *Journal of Clinical Hypertension*.

Murray J, Bernacchia A, Roy N, Constantinidis C, Romo R, Wang X, Fusi S, Martinez-Trujillo J (2017a). “Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex.” *Proc. Natl. Acad. Sci*.

Murray J, Jaramillo J, Wang X (2017b). “Working Memory and Decision Making in a Frontoparietal Circuit Model.” *J Neurosci*.

Peters A, Sethares C, Luebke JI (2008). “Synapses are lost during aging in the primate prefrontal cortex.” *Neuroscience*.

Possin K, Filoteo J, Song D, Salmon D (2008). "Spatial and Object Working Memory Deficits in Parkinson's Disease are Due to Impairment in Different Underlying Processes." *Neuropsychology.*

Prasad S, Prasad P (2014). "Deep Recurrent Neural Networks for Time- Series Prediction." *arXiv.*

Rajalingham R, Issa E, Bashivan P, Kar K, Schmidt K, Dicarlo J (2018). "Large-Scale , High-Resolution Comparison of the Core Visual Object Recognition Behavior of Humans , Monkeys , and State-of-the-Art Deep Artificial Neural Networks." *J. Neurosci.*

Raven J (1962). *Advanced progressive matrices.* Lewis and Co Ltd.

Roberts L (1963). "Machine Perception of Three-Dimensional Solids." *Massachusetts Institute of Technology.*

Romero A, Ballas N, Kahou S, Chassang A (2015). "FitNets: Hints for Thin Deep Nets." *ICLR.*

Romo R, Brody CD, Hernández a, Lemus L (1999). "Neuronal correlates of parametric working memory in the prefrontal cortex." *Nature.*

Rosenblatt F (1958). "The Perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review.*

Rumelhart D, Hinton G, Williams R (1986a). "Learning representations by back-propagating errors." *Nature.*

Rumelhart D, Hinton G, Williams R (1986b). "Learning representations by back-propagating errors." *Nature.*

Schewizer K, Moosbrugger H (2004). "Attention and working memory as predictors of intelligence." *Intelligence.*

Schmidhuber J (1993). "Habilitation thesis: System modeling and optimization." *Technische Universitat Munchen.*

Seung S, Lee D, Reis B, Tank D (2000). "The Autapse : A Simple Illustration of Short-Term Analog Memory Storage by Tuned Synaptic Feedback." *J Comput Neurosci.*

Seymour P (1966). "The Summer Vision Project." *MIT AI Memos.*

Simonyan K, Zisserman A (2015). "Very deep convolutional networks for large-scale image recognition." *ICLR.*

Skogan A, Egeland J, Rohrer-Baumgartner N, Unres A, Reichborn-Kjennerud T, Aase H (2014). "Inhibition and working memory in young preschool children with symptoms of ADHD and/or oppositional-defiant disorder." *Child Neuropsychol.*

Soriano MC (2017). "Viewpoint: Reservoir Computing Speeds Up." *Physics 10, 12.*

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *J Mach Learn Res.*

Sternberg R, Berg C (1986). "What is intelligence?" *Ablex.*

Sussillo D, Abbott LF (2009). "Generating Coherent Patterns of Activity from Chaotic Neural Networks." *Neuron.*

Tan M, Le Q (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." *ICML.*

Toyoizumi T, Abbott LF (2011). "Beyond the edge of chaos: Amplification and temporal integration by recurrent networks in the chaotic regime." *Phys Rev E Stat Nonlin Soft Matter Phys.*

Turing AM (1950). "Computing machinery and intelligence." *Mind.*

van der Maaten L, Hinton G (2008). "Visualizing Data using t-SNE." *J. Mach. Learn. Res.*

Vertechi P, Machens CK (2014). "Unsupervised learning of an efficient short-term memory network." *Adv. Neural Inf. Process. Syst.*

Wimmer K, Nykamp D, Constantinidis C, Compte A (2014). “Bump attractor dynamics in prefrontal cortex explains behavioral precision in spatial working memory.” *Nat. Neurosci.*

Xie X, Seung HS (2000). “Spike-based learning rules and stabilization of persistent neural activity.” *Adv. Neural Inf. Process. Syst.*

Yamins D, Dicarlo J (2016). “Using goal-driven deep learning models to understand sensory cortex.” *Nat. Neuro.*

Yamins D, Hong H, Cadieu C, Solomon E, Seibert D, Dicarlo J (2014). “Performance-optimized hierarchical models predict neural responses in higher visual cortex.” *PNAS.*

Yosinski J, Clune J, Bengio Y, Lipson H (2014). “How transferable are features in deep neural networks?” *NIPS.*

Zdziarski Z (2018). “The Early History of Computer Vision.” *zbigatron.*

Zhu X, Bain M (2017). “B-CNN: Branch Convolutional Neural Network for Hierarchical Classification.” *arXiv.*

Zylberberg J, Murphy J, DeWeese M (2011). “A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields.” *PLoS Comput. Biol.*

Zylberberg J, Strowbridge BW (2017). “Mechanisms of Persistent Activity in Cortical Circuits : Possible Neural Substrates for Working Memory.” *Annu. Rev. Neurosci.*