CellPress Neuron

Primer

Artificial Neural Networks for Neuroscientists: A Primer

Guangyu Robert Yang[1,*] and Xiao-Jing Wang[2,*]
[1]Center for Theoretical Neuroscience, Columbia University, New York, NY, USA
[2]Center for Neural Science, New York University, New York, NY, USA
*Correspondence: robert.yang@columbia.edu (G.R.Y.), xjwang@nyu.edu (X.-J.W.)
https://doi.org/10.1016/j.neuron.2020.09.005

SUMMARY

Artificial neural networks (ANNs) are essential tools in machine learning that have drawn increasing attention in neuroscience. Besides offering powerful techniques for data analysis, ANNs provide a new approach for neuroscientists to build models for complex behaviors, heterogeneous neural activity, and circuit connectivity, as well as to explore optimization in neural systems, in ways that traditional models are not designed for. In this pedagogical Primer, we introduce ANNs and demonstrate how they have been fruitfully deployed to study neuroscientific questions. We first discuss basic concepts and methods of ANNs. Then, with a focus on bringing this mathematical framework closer to neurobiology, we detail how to customize the analysis, structure, and learning of ANNs to better address a wide range of challenges in brain research. To help readers garner hands-on experience, this Primer is accompanied with tutorial-style code in PyTorch and Jupyter Notebook, covering major topics.

1. ARTIFICIAL NEURAL NETWORKS IN NEUROSCIENCE

Learning with artificial neural networks (ANNs), or deep learning, has emerged as a dominant framework in machine learning nowadays (LeCun et al., 2015), leading to breakthroughs across a wide range of applications, including computer vision (Krizhevsky et al., 2012), natural language processing (Devlin et al., 2018), and strategic games (Silver et al., 2017). Some key ideas in this field can be traced to brain research: supervised learning rules have their roots in the theory of training perceptrons, which, in turn, was inspired by the brain (Rosenblatt, 1962); the hierarchical architecture (Fukushima and Miyake, 1982) and convolutional principle (LeCun and Bengio, 1995) were closely linked to our knowledge about the primate visual system (Hubel and Wiesel, 1962; Felleman and Van Essen, 1991). Today, there is a continued exchange of ideas from neuroscience to the field of artificial intelligence (Hassabis et al., 2017).

At the same time, machine learning offers new and powerful tools for systems neuroscience. One utility of the deep learning framework is to analyze neuroscientific data (Figure 1). Indeed, the advances in computer vision, especially convolutional neural networks, have revolutionized image and video data processing. For instance, uncontrolled behaviors over time, such as micro-movements of animals in a laboratory experiment, can now be tracked and quantified efficiently with the help of deep neural networks (Mathis et al., 2018). Innovative neurotechnologies are producing a deluge of big data from brain connectomics, transcriptome, and neurophysiology, the analyses of which can benefit from machine learning. Examples include image segmentation to achieve detailed, micrometer scale, reconstruction of connectivity in a neural microcircuit (Januszewski et al., 2018; Helmstaedter et al., 2013), and estimation of neural firing rate from spiking data (Pandarinath et al., 2018).

This Primer will not be focused on data analysis; instead, our primary aim is to present basic concepts and methods for the development of ANN models of biological neural circuits in the field of computational neuroscience. It is noteworthy that ANNs should not be confused with neural network models in general. Mathematical models are all “artificial” because they are not biological. We denote by ANNs specifically models that are in part inspired by neuroscience yet for which biologically justification is not the primary concern, in contrast to other types of models that strive to be built on quantitative data from the two pillars of neuroscience: neuroanatomy and neurophysiology. The use of ANNs in neuroscience (Zipser and Andersen, 1988) and cognitive science (Cohen et al., 1990) dates back to the early days of ANNs (Rumelhart et al., 1986). In recent years, ANNs are becoming increasingly common model systems in neuroscience (Yamins and DiCarlo, 2016; Kriegeskorte, 2015; Sussillo, 2014; Barak, 2017). There are three reasons for which ANNs or deep learning models have already been, and will likely continue to be, particularly useful for neuroscientists.

First, fresh modeling approaches are needed to meet new challenges in brain research. Over the past decades, computational neuroscience has made great strides and become an integrated part of systems neuroscience (Abbott, 2008). Many insights have been gained through integration of experiments and theory, including the idea of excitation and inhibition balance (van Vreeswijk and Sompolinsky, 1996; Shu et al., 2003) and normalization (Carandini and Heeger, 2011). Progress was also made in developing models of basic cognitive functions, such as simple decision making (Gold and Shadlen, 2007; Wang, 2008). However, real-life problems can be incredibly complex;

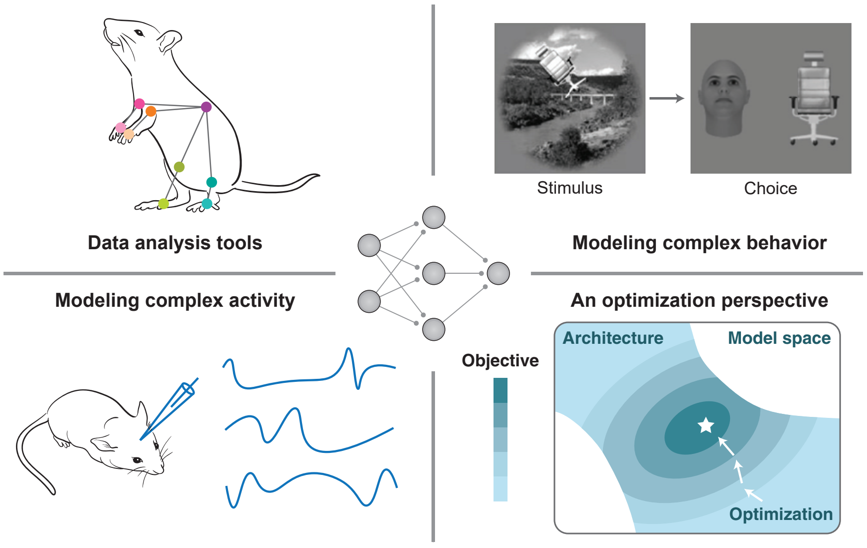
Stimulus
Choice
Data analysis tools
Modeling complex behavior
Modeling complex activity
An optimization perspective
Objective
Architecture
Model space
Optimization

Figure 1. Reasons for Using ANNs for Neuroscience Research

(Top left) Neural/Behavioral data analysis. ANNs can serve as image processing tools for efficient pose estimation (color dots). Figure inspired by Nath et al. (2019).

(Top right) Modeling complex behaviors. ANNs can perform object discrimination tasks involving challenging naturalistic visual objects. Figure adapted from Kar et al. (2019).

(Bottom left) Illustrating that ANNs can be used to model complex neural activity/connectivity patterns (blue lines).

(Bottom right) Understanding neural circuits from an optimization perspective. In this view, functional neural networks (star symbol) are results of the optimization (arrows) of an objective function in an abstract space of a model constrained by the neural network architecture (colored space).

the underlying brain systems are often difficult to capture with "hand-constructed" computational models. For example, object classification in the brain is carried out through many layers of complex linear-nonlinear processing. Building functional models of the visual systems that achieve behavioral performance close to that of humans remained a formidable challenge not only for neuroscientists but also for computer vision researchers. By directly training neural network models on complex tasks and behaviors, deep learning provides a way to efficiently generate candidate models for brain functions that otherwise could be near impossible to model (Figure 1). By learning to perform a variety of complex behaviors of animals, ANNs could serve as potential model systems for biological neural networks, complementing nonhuman animal models for understanding the human brain.

A second reason for advocating deep networks in systems neuroscience is the acknowledgment that relatively simple models often do not account for a wide diversity of activity patterns in heterogeneous neural populations (Figure 1). One can rightly argue that this is a virtue rather than a defect because simplicity and generality are hallmarks of good theories. However, complex neural signals also tell us that existing models may be insufficient to elucidate mysteries of the brain. This is perhaps especially true in the case of the prefrontal cortex. Neurons in prefrontal cortex often show complex mixed selectivity to various task variables (Rigotti et al., 2010, 2013). Such complex patterns are often not straightforward to interpret and understand using hand-built models that by design strive for simplicity. ANNs are promising to capture the complex nature of neural activity.

Third, besides providing mechanistic models of biological systems, machine learning can be used to probe the "why" question in neuroscience (Barlow, 1961). Brains are biological machines evolved under pressure to compute robustly and efficiently. Even when we understand how a system works, we may still ask *why* it works that way. Similar to biological systems evolving to survive, ANNs are trained to optimize objective functions given various architectural constraints (the number of neurons, economy of circuit wiring, etc.) (Figure 1). By identifying the particular objective and set of constraints that lead to brain-resembling ANNs, we could potentially gain insights into the evolutionary pressure faced by biological systems (Richards et al., 2019).

In this pedagogical Primer, we will discuss how ANNs can benefit neuroscientists in the three ways described above. In section 2, we will first introduce the key ingredients common in any study of ANNs. In section 3, we will describe two major applications of ANNs as neuroscientific models: convolutional networks as models for sensory, especially visual, systems and recurrent neural networks as models for cognitive and motor systems. In sections 4 and 5, we will overview how to customize the analysis and architectural design of ANNs to better address a wide range of neuroscience questions. To help the readers gain hands-on experience, we accompany this Primer with tutorial-style code in PyTorch and Jupyter Notebook (https://github.com/gyyang/nn-brain), covering all major topics.

2. BASIC INGREDIENTS AND VARIATIONS IN ANNs

In this section, we will introduce basic concepts in ANNs and their common variations. Readers can skip this section if they are familiar with ANNs and deep learning. For a more thorough introduction, readers can refer to Goodfellow et al. (2016).

2.1. Basic Ingredient: Learning Problem, Architecture, and Algorithm

A typical study using deep networks consists of three basic ingredients: learning problem, network architecture, and training algorithm. Weights of connections between units or neurons in a neural network are constrained by the network architecture, but their specific values are randomly assigned at initialization. These weights constitute a large number of parameters, collectively denoted by θ, which also includes other model parameters (see below), to be trained using an algorithm. The training algorithm specifies how connection weights change to better solve

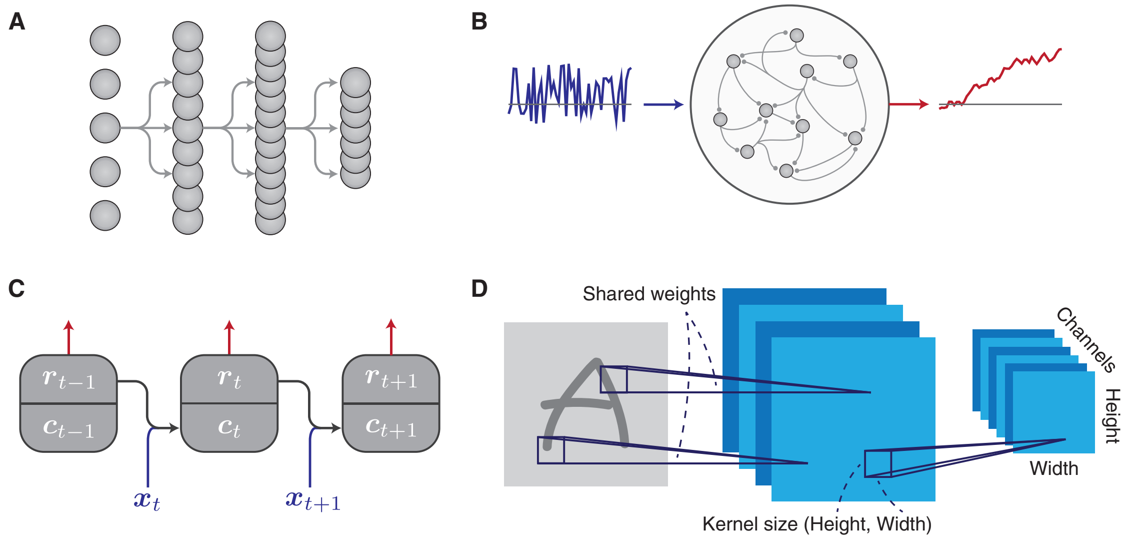

Figure 2. Schematics of Common Neural Network Architectures

(A) A multi-layer perceptron (MLP).

(B) A recurrent neural network (middle) receives a stream of inputs (left). After training, an output unit (right) should produce a desired output. Figure inspired by Mante et al. (2013).

(C) A recurrent neural network is unrolled in time as a feedforward system with each layer corresponding to the network state at one time step. $\boldsymbol{c}_t$ and $\boldsymbol{r}_t$ describe the network state and output activity at time t, respectively. $\boldsymbol{c}_t$ is a function of $\boldsymbol{r}_{t-1}$ and the input $\boldsymbol{x}_t$.

(D) A convolutional neural network for processing images. Each layer contains a number of channels (four in layer 1, six in layer 2). A channel (represented by a square) consists of spatially organized neurons, each receiving connections from neurons with similar spatial preferences. The spatial extent of these connections is described by the kernel size. Figure inspired by LeCun et al. (1998).

a learning problem, such as to fit a dataset or perform a task. We will go over a simple example in which a multi-layer perceptron (MLP) is trained to perform a simple digit classification task using supervised learning.

Learning Problem

In supervised learning, a system learns to fit a dataset containing a set of inputs $\{\boldsymbol{x}^{(i)}\}, i = 1, \cdots, N$. Each input $\boldsymbol{x}^{(i)}$ is paired with a target output $\boldsymbol{y}^{(i)}_{\text{target}}$. Symbols in bold represent vectors (column vectors by default). The goal is to learn parameters $\boldsymbol{\theta}$ of a neural network function $F(\cdot, \boldsymbol{\theta})$ that predicts the target outputs given inputs, $\boldsymbol{y}^{(i)} = F(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \approx \boldsymbol{y}^{(i)}_{\text{target}}$. In the simple digit-classification task MNIST (LeCun et al., 1998), each input is an image containing a single digit, while the target output is a probability distribution over all classes (0, 1, …, 9) given by a ten-dimensional vector or simply an integer corresponding to the class of that object.

More precisely, the system is trained to optimize the value of an objective function or, commonly, minimize the value of a loss function $L = \frac{1}{N}\sum_i L\left(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(i)}_{\text{target}}\right)$, where $L\left(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(i)}_{\text{target}}\right)$ quantifies the difference between the target output $\boldsymbol{y}^{(i)}_{\text{target}}$ and the actual output $\boldsymbol{y}^{(i)}$.

Network Architecture

ANNs are incredibly versatile, including a wide range of architectures. Of all architectures, the most fundamental one is an MLP (Rosenblatt, 1958, 1962) (Figure 2A). An MLP consists of multiple layers of neurons, where neurons in the l-th layer only receive inputs from the $(l-1)$-th layer and only project to the $(l+1)$-th layer.

$$\boldsymbol{r}^{(1)} = \boldsymbol{x}, \qquad \text{(Equation 1)}$$

$$\boldsymbol{r}^{(l)} = f\left(\boldsymbol{W}^{(l)}\boldsymbol{r}^{(l-1)} + \boldsymbol{b}^{(l)}\right), \; 1 < l < N, \qquad \text{(Equation 2)}$$

$$\boldsymbol{y} = \boldsymbol{W}^{(N)}\boldsymbol{r}^{(N-1)} + \boldsymbol{b}^{(N)}. \qquad \text{(Equation 3)}$$

Here, $\boldsymbol{x}$ is an external input, $\boldsymbol{r}^{(l)}$ denotes the neural activity of neurons in the l-th layer, and $\boldsymbol{W}^{(l)}$ is the connection matrix from the $(l-1)$-th to the l-th layer. $f(\cdot)$ is a (usually nonlinear) activation function of the model neurons. The output of the network is read out through connections $\boldsymbol{W}^{(N)}$. Parameters $\boldsymbol{b}^{(l)}$ and $\boldsymbol{b}^{(N)}$ are biases for model neurons and output units, respectively. If the network is trained to classify, then the output is often normalized such that $\sum_j y_j = 1$, where y_j represents the predicted probability of class j.

When there are enough neurons per layer, MLPs can, in theory, approximate arbitrary functions (Hornik et al., 1989). However, in practice, the network size is limited, and good solutions may not be found through training even when they exist. MLPs are often used in combination with, or as parts of, more modern neural network architectures.

Training Algorithm

The signature method of training in deep learning is stochastic gradient descent (SGD) (Robbins and Monro, 1951; Rumelhart et al., 1986). Trainable parameters, collectively denoted as $\boldsymbol{\theta}$, are updated in the opposite direction of the gradient of the loss, $\partial L / \partial \boldsymbol{\theta}$. Intuitively, the j-th parameter θ_j should be reduced by training if the cost function L increases with it and increased otherwise. For each step of training, because it is usually too expensive to evaluate the loss using the entire training set, the loss is computed using a small number M of randomly selected training examples (a minibatch), indexed by $\mathbb{B} = \{k_1, \cdots, k_M\}$,

$$L_{\text{batch}} = \frac{1}{M}\sum_{k\in\mathbb{B}} L\left(\boldsymbol{y}^{(k)}, \boldsymbol{y}^{(k)}_{\text{target}}\right), \qquad \text{(Equation 4)}$$

hence the name "stochastic." For simplicity, we assume a minibatch size of 1 and omit batch in the following equations (L_{batch} will be referred to as L, etc.). The gradient, $\partial L/\partial\boldsymbol{\theta}$, is the direction of parameter change that would lead to the maximum increase in the loss function when the change is small enough. To decrease the loss, trainable parameters are updated in the opposite direction of the gradient, with a magnitude proportional to the learning rate η,

$$\Delta\boldsymbol{\theta} = -\eta\frac{\partial L}{\partial\boldsymbol{\theta}}. \qquad \text{(Equation 5)}$$

Parameters such as $\boldsymbol{W}$ and $\boldsymbol{b}$ are usually trainable. Other parameters are set by the modelers and called hyperparameters, for example, the learning rate η. A crucial requirement for computing gradients is differentiability—namely, derivatives of functions in the model are well defined.

For a feedforward network without any intermediate (hidden) layer (Rosenblatt, 1962), processing a single example $\boldsymbol{x}$ (minibatch size 1),

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}, \quad \text{or equivalently,} \quad y_i = \sum_j W_{ij}x_j + b_i, \qquad \text{(Equation 6)}$$

computing the gradient is straightforward,

$$\frac{\partial L}{\partial W_{ij}} = \sum_k \frac{\partial L}{\partial y_k}\frac{\partial y_k}{\partial W_{ij}} = \frac{\partial L}{\partial y_i}x_j, \qquad \text{(Equation 7)}$$

with $\partial y_k/\partial W_{ij}$ equal to x_j when $k = i$, otherwise 0. In vector notation,

$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{\partial L}{\partial \boldsymbol{y}}\boldsymbol{x}^{\top}. \qquad \text{(Equation 8)}$$

Here, we follow the convention that $\partial L/\partial\boldsymbol{W}$ and $\partial L/\partial\boldsymbol{y}$ have the same form as $\boldsymbol{W}$ and $\boldsymbol{y}$, respectively. Assuming that

$$L = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{y}_{\text{target}}\|^2 = \frac{1}{2}\sum_j\left(y_j - y_{\text{target},j}\right)^2, \qquad \text{(Equation 9)}$$

we have,

$$\frac{\partial L}{\partial \boldsymbol{W}} = \left(\boldsymbol{y} - \boldsymbol{y}_{\text{target}}\right)\boldsymbol{x}^{\top}, \qquad \text{(Equation 10)}$$

$$\Delta W_{ij} \propto -\frac{\partial L}{\partial W_{ij}} = \left(y_{\text{target},i} - y_i\right)x_j. \qquad \text{(Equation 11)}$$

This modification only depends on local information about the input and output units of each connection. Hence, if $y_{\text{target},i} > y_i$, W_{ij} should change to increase the net input and ΔW_{ij} has the same sign as x_j. The opposite is true if $y_{\text{target},i} < y_i$.

For a multi-layer network, the differentiation is done using the back-propagation algorithm (Rumelhart et al., 1986; LeCun, 1988). To compute the loss L, the network is run in a forward pass (Equations 1, 2, and 3). Next, to efficiently compute the exact gradient $\partial L/\partial\boldsymbol{\theta}$, information about the loss needs to be passed backward, in the opposite direction of the forward pass, hence the name backpropagation.

To illustrate the concept, consider an N-layer linear feedforward network (Equations 1, 2, and 3, but with $f(\boldsymbol{x}) = \boldsymbol{x}$). To compute $\partial L/\partial\boldsymbol{W}^{(l)}$, we need to compute $\partial L/\partial\boldsymbol{r}^{(l)}$. From $\boldsymbol{r}^{(l+1)} = \boldsymbol{W}^{(l+1)}\boldsymbol{r}^{(l)} + \boldsymbol{b}^{(l+1)}$, we have

$$\frac{\partial L}{\partial r_i^{(l)}} = \sum_j \frac{\partial L}{\partial r_j^{(l+1)}}\frac{\partial r_j^{(l+1)}}{\partial r_i^{(l)}} = \sum_j \frac{\partial L}{\partial r_j^{(l+1)}}W_{ji}^{(l+1)} = \sum_j \left[W^{(l+1)}\right]^{\top}_{ij}\frac{\partial L}{\partial r_j^{(l+1)}}. \qquad \text{(Equation 12)}$$

In vector notation,

$$\frac{\partial L}{\partial \boldsymbol{r}^{(l)}} = \left[\boldsymbol{W}^{(l+1)}\right]^{\top}\frac{\partial L}{\partial \boldsymbol{r}^{(l+1)}} = \left[\boldsymbol{W}^{(l+1)}\right]^{\top}\left[\boldsymbol{W}^{(l+2)}\right]^{\top}\frac{\partial L}{\partial \boldsymbol{r}^{(l+2)}} = \cdots. \qquad \text{(Equation 13)}$$

Therefore, starting with $\partial L/\partial\boldsymbol{y}$, $\partial L/\partial\boldsymbol{r}^{(l)}$ can be recursively computed from $\partial L/\partial\boldsymbol{r}^{(l+1)}$, for $l = N-1, \cdots, 1$. This computation flows in the opposite direction of the forward pass and is called the backward pass. In general, backpropagation applies to neural networks with arbitrary differential components.

Computing the exact gradient through backpropagation is considered unrealistic biologically because updating connections at each layer requires precise, non-local information of connection weights at downstream layers (in the form of connection matrix transposed, Equation 13).

2.2. Variations of Learning Problems/Objective Functions

In this and the following sections (2.3 and 2.4), we introduce common variations of learning problems, network architectures, and training algorithms.

Traditionally, learning problems are divided into three kinds: supervised, reinforcement, and unsupervised learning problems. The difference across these three kinds of learning problems lies in the goal or objective. In supervised learning, each input is associated with a target. The system learns to produce outputs that match the targets. In reinforcement learning, instead of explicit (high-dimensional) targets, the system receives a series of scalar rewards. It learns to produce outputs (actions) that maximize total rewards. Unsupervised learning refers to a diverse set of problems in which the system is not provided with explicit targets or rewards. Due to space limitations, we will mainly focus on networks trained with supervised learning in this Primer.

Supervised Learning

As mentioned before, for supervised learning tasks, input and target output pairs are provided $\left\{\left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}_{\text{target}}\right)\right\}$. The goal is to minimize the difference between target outputs and actual outputs predicted by the network. In many common supervised learning problems, the target outputs are behavioral outputs. For example, in a typical object classification task, each input is an image containing a single object, while the target output is an integer corresponding to the class of that object (e.g., dog, cat, etc.). In other cases, the target output can directly be neural recording data (McIntosh et al., 2016; Rajan et al., 2016; Andalman et al., 2019).

The classical perceptual decision-making task with random-dot motion (Britten et al., 1992; Roitman and Shadlen, 2002) can be formulated as a supervised learning problem because there is a correct answer. In this task, animals watch randomly moving dots and report the dots' overall motion direction by choosing one of two alternatives, A or B. This task can be simplified as a network receiving a stream of noisy inputs $x_t^{(i)}$ at every time point t of the i-th trial, which can represent the net evidence in support of A and against B. At the end of each trial $t = T$, the system should learn to report the sign of the average input $y_{\text{target}}^{(i)} = \text{sign}(\langle x_t^{(i)} \rangle_t)$, $+1$ for choice A and -1 for choice B.

Reinforcement Learning

For reinforcement learning (Sutton and Barto, 2018), a model (an agent) interacts with an environment, such as a (virtual) maze. At time step t, the agent receives an observation $\boldsymbol{o}_t$ from the environment, produces an action a_t that updates the environment state to $\boldsymbol{s}_{t+1}$, and receives a scalar reward r_t (negative value for punishment). For example, a model navigating a virtual maze can receive pixel-based visual inputs as observations $\boldsymbol{o}_t$, produce actions a_t that move itself in the maze, and receive rewards when it exits the maze. The objective is to produce appropriate actions a_t given past and present observations that maximize cumulative rewards $\sum_t r_t$. In many classical reinforcement learning problems, the observation $\boldsymbol{o}_t$ equals the environment state $\boldsymbol{s}_t$, which contains complete information about the environment.

Reinforcement learning (without neural networks) has been widely used by neuroscientists and cognitive scientists to study value-based learning and decision-making tasks (Schultz et al., 1997; Daw et al., 2011; Niv, 2009). For example, in the multi-armed bandit task, the agent chooses between multiple options repeatedly, where each option produces rewards with a certain probability. Reinforcement learning theory can model how the agent's behavior adapts over time and help neuroscientists study the neural mechanism of value-based behavior.

Deep reinforcement learning trains deep neural networks using reinforcement learning (Mnih et al., 2015), enabling applications to many more complex problems. Deep reinforcement learning can, in principle, be used to study most tasks performed by lab animals (Botvinick et al., 2020) because animals are usually motivated to perform the task via rewards. Although many such tasks can also be formulated as supervised learning problems when there exists a correct choice (e.g., perceptual decision making), many other tasks can only be described as reinforcement learning tasks because answers are subjective (Haroush and Williams, 2015; Kiani and Shadlen, 2009). For example, a perceptual decision-making task in which there is a correct answer (A, not B) can be extended to assess animals' confidence about their choice (Kiani and Shadlen, 2009; Song et al., 2017). In addition to the two alternatives that result in a large reward for the correct choice and no reward otherwise, monkeys are presented a sure-bet option that guarantees a small reward. Since a small reward is better than no reward, subjects are more likely to choose the sure-bet option when they are less confident about making a perceptual judgement. Reinforcement learning is necessary here because there is no ground-truth choice output: the optimal choice depends on the animals' own confidence level at their perceptual decision.

Unsupervised Learning

For unsupervised learning, only inputs $\{\boldsymbol{x}^{(i)}\}$ are provided; the objective function is defined solely with the inputs and the network parameters $L(\boldsymbol{x}, \boldsymbol{\theta})$ (no targets or rewards). For example, finding the first component in principal-component analysis (PCA) can be formulated as unsupervised learning in a simple neural network. A single neuron y reading out from a group of input neurons $\boldsymbol{x}$, ($\boldsymbol{y} = \boldsymbol{w}^\top \boldsymbol{x}$), can learn to extract the first principle component by maximizing its variance $\text{Var}(y)$ while keeping its connection weights normalized ($||\boldsymbol{w}|| = 1$) (Oja, 1982).

Unsupervised learning is particularly relevant for modeling development of sensory cortices. Although widely used in machine learning, the kind of labeled data needed for supervised learning, such as image-object class pairs, is rare for most animals. Unsupervised learning has been used to explain neural responses of early visual areas (Barlow, 1961; Olshausen and Field, 1996) and, more recently, of higher visual areas (Zhuang et al., 2019).

Compared to reinforcement and unsupervised learning, supervised learning can be particularly effective because the network receives more informative feedback in the form of high-dimensional target outputs. Therefore, it is common to formulate a reinforcement/unsupervised learning problem (or parts of it) as a supervised one. For example, consider an unsupervised learning problem of compressing high-dimensional inputs $\boldsymbol{x}$ into lower-dimensional representation $\boldsymbol{z}$ while retaining as much information as possible about the inputs (not necessarily in the information-theoretic sense). One approach to this problem is to train autoencoder networks (Rumelhart et al., 1986; Kingma and Welling, 2013) using supervised learning. An autoencoder consists of an encoder that maps input $\boldsymbol{x}$ into a low-dimensional latent representation $\boldsymbol{z} = f_{\text{encode}}(\boldsymbol{x})$ and a decoder that maps the latent back to a high-dimensional representation $\boldsymbol{y} = f_{\text{decode}}(\boldsymbol{z})$. To make sure $\boldsymbol{z}$ contains information about $\boldsymbol{x}$, autoencoders use the original input as the supervised learning target, $\boldsymbol{y}_{\text{target}} = \boldsymbol{x}$.

2.3. Variations of Network Architectures

Recurrent Neural Network

Besides MLP, another fundamental ANN architecture is recurrent neural networks (RNNs) that process information in time (Figure 2B). In a "vanilla" or Elman RNN (Elman, 1990), activity of model neurons at time t, $\boldsymbol{r}_t$, is driven by recurrent connectivity $\boldsymbol{W}_r$ and by inputs $\boldsymbol{x}_t$ through connectivity $\boldsymbol{W}_x$. The output of the network is read out through connections $\boldsymbol{W}_y$.

$$\boldsymbol{c}_t = \boldsymbol{W}_r \boldsymbol{r}_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{b}_r, \qquad \text{(Equation 14)}$$

$$\mathbf{r}_t = f(\mathbf{c}_t), \tag{Equation 15}$$

$$\mathbf{y}_t = \mathbf{W}_y \mathbf{r}_t + \mathbf{b}_y. \tag{Equation 16}$$

Here, $\mathbf{c}_t$ represents the cell state, analogous to membrane potential or input current, while $\mathbf{r}_t$ represents the neuronal activity. An RNN can be unrolled in time (Figure 2C) and viewed as a particular form of an MLP,

$$\mathbf{r}_t = f(\mathbf{W}_r \mathbf{r}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}_r), \quad \text{for } t = 1, \cdots, T. \tag{Equation 17}$$

Here, neurons in the t-th layer $\mathbf{r}_t$ receive inputs from the $(t-1)$-th layer $\mathbf{r}_{t-1}$ and additional inputs from outside of the recurrent network $\mathbf{x}_t$. Unlike regular MLPs, the connections from each layer to the next are shared across time.

Backpropagation also applies to an RNN. While backpropagation in an MLP propagates gradient information from the final layer back (Equation 13), computing the gradient for an RNN involves propagating information backward in time (backpropagation-through-time, or BPTT) (Werbos, 1990). Assuming that the loss is computed from outputs at the last time point T and a linear activation function, the key step of BPTT is computed similarly to Equation 13 as

$$\frac{\partial L}{\partial \mathbf{r}_t} = \mathbf{W}_r^\top \frac{\partial L}{\partial \mathbf{r}_{t+1}} = \left[\mathbf{W}_r^\top\right]^2 \frac{\partial L}{\partial \mathbf{r}_{t+2}} = \cdots. \tag{Equation 18}$$

With an increasing number of time steps in an RNN, weight modifications involve products of many matrices (Equation 18). An analogous problem is present for very deep feedforward networks (for example, networks with more than ten layers). The norm of this matrix product, $\|\left[\mathbf{W}_r^\top\right]^T\|$, can grow exponentially with T if $\mathbf{W}_r$ is large (more precisely, the largest eigenvalue of $\mathbf{W}_r > 1$) or vanish to zero if $\mathbf{W}_r$ is small, making it historically difficult to train recurrent networks (Bengio et al., 1994; Pascanu et al., 2013). Such exploding and vanishing gradient problems can be substantially alleviated with a combination of modern techniques, including network architectures (Hochreiter and Schmidhuber, 1997; He et al., 2016) and initial network connectivity (Le et al., 2015; He et al., 2015) that tend to preserve the norm of the backpropagated gradient.

Convolutional Neural Networks

A particularly important type of network architecture is convolutional neural network (Figure 2D). The use of convolution means that a group of neurons will each process its respective inputs using the same function—in other words, the same set of connection weights. In a typical convolutional neural network processing visual inputs (Fukushima et al., 1983; LeCun et al., 1990; Krizhevsky et al., 2012; He et al., 2016), neurons are organized into N_{channel} "channels" or "feature maps." Each channel contains $N_{\text{height}} \times N_{\text{width}}$ neurons with different spatial selectivity. Each neuron in a convolutional layer is indexed by a tuple $i = (i_C, i_H, i_W)$, representing the channel index (i_C) and the spatial preference indices (i_H, i_W). The i-th neuron in layer l is typically driven by neurons in the previous layer (bias term and activation function omitted),

$$r^{(l)}_{i_C i_H i_W} = \sum_{j_C j_H j_W} W^{(l)}_{i_C i_H i_W, j_C j_H j_W} r^{(l-1)}_{j_C j_H j_W}. \tag{Equation 19}$$

Importantly, in convolutional networks, the connection weights do not depend on the absolute spatial location of the i-th neuron; instead, they depend solely on the spatial displacement $(i_H - j_H, i_W - j_W)$ between the pre- and postsynaptic neurons.

$$W^{(l)}_{i_C i_H i_W, j_C j_H j_W} = W^{(l)}_{i_C j_C}(i_H - j_H, i_W - j_W). \tag{Equation 20}$$

Therefore, all neurons within a single channel process different parts of the input space using the same shared set of connection weights, allowing these neurons to have the same stimulus selectivity with receptive fields at different spatial locations. Moreover, neurons only receive inputs from other neurons with similar spatial preferences, i.e., when $|i_H - j_H|$ and $|i_W - j_W|$ values are small (Figure 2D).

This reusing of weights not only dramatically reduces the number of trainable parameters but also imposes invariance on processing. For visual processing, convolutional networks typically impose spatial invariance such that objects are processed with the same set of weights regardless of their spatial positions.

In a typical convolutional network, across layers, the number of neurons per channel $(N_{\text{height}} \times N_{\text{width}})$ decreases (with coarser spatial resolution) while more features are extracted (with an increasing number of channels). A classifier is commonly at the end of the system to learn a particular task, such as categorization of visual objects.

Activation Function

Most neurons in ANNs, like their biological counterparts, perform nonlinear computations based on their inputs. These neurons are usually point neurons with a single nonlinear activation function $f(\cdot)$ that links the sum of inputs to the output activity. The nonlinearity is essential for the power of ANNs (Hornik et al., 1989). A common choice of activation function is the Rectified Linear Unit (ReLU) function, $f(x) = \max(x, 0)$ (Glorot et al., 2011). The derivative of ReLU at $x = 0$ is mathematically undefined but conventionally set to 0 in practice. ReLU and its variants (Clevert et al., 2015) are routinely used in feedforward networks, while the hyperbolic tangent (tanh) function is often used in recurrent networks (Hochreiter and Schmidhuber, 1997). ReLU and similar activation functions are asymmetric and non-saturating at high value. Although biological neurons eventually saturate at high rate, they often operate in non-saturating regimes. Therefore, traditional neural circuit models with rate units have also frequently used non-saturating activation functions (Abbott and Chance, 2005; Rubin et al., 2015).

Normalization

Normalization methods are important components of many ANNs, in particular, very deep neural networks (Ioffe and Szegedy, 2015; Ba et al., 2016b; Wu and He, 2018). Similar to normalization in biological neural circuits (Carandini and Heeger, 2011), normalization methods in ANNs keep inputs and/or outputs of neurons in desirable ranges. For example, for inputs $\mathbf{x}$ (e.g., stimulus) to a layer, layer normalization (Ba et al., 2016b) amounts to a form of "Z scoring" across units, so that the actual input $\widehat{x}_i$ to the i-th neuron is

$$\widehat{x}_i = \gamma \cdot \frac{x_i - \mu}{\sigma} + \beta, \qquad \text{(Equation 21)}$$

$$\mu = \langle x_j \rangle, \qquad \text{(Equation 22)}$$

$$\sigma = \sqrt{\langle (x_j - \mu)^2 \rangle + \epsilon}. \qquad \text{(Equation 23)}$$

where $\langle x_j \rangle$ refers to the average over all units in the same layer; μ and σ are the mean and variance of $\boldsymbol{x}$. After normalization, different external inputs lead to the same mean and variance for $\widehat{\mathbf{x}}$, set by the trainable parameters γ and β. The values of γ and β do not depend on the external inputs. The small constant ε ensures that σ is not vanishingly small.

2.4. Variations of Training Algorithms

Variants of SGD-Based Methods

Supervised, reinforcement, and unsupervised learning tasks can all be trained with SGD-based methods. Partly due to the stochastic nature of the estimated gradient, directly applying SGD (Equation 5) often leads to poor training performance. Gradually decaying learning rate value η during training can often improve performance, because a smaller learning rate during late training encourages finer-tuning of parameters (Bottou et al., 2018). Various optimization methods based on SGD are used to improve learning (Kingma and Ba, 2014; Sutskever et al., 2013). One simple and effective technique is momentum (Sutskever et al., 2013; Polyak, 1964), which on step j updates parameters with $\Delta\boldsymbol{\theta}^{(j)}$ based on temporally smoothed gradients $\boldsymbol{v}^{(j)}$,

$$\boldsymbol{v}^{(j)} = \mu \boldsymbol{v}^{(j-1)} + \frac{\partial L^{(j)}}{\partial \boldsymbol{\theta}}, \quad 0 < \mu < 1 \qquad \text{(Equation 24)}$$

$$\Delta\boldsymbol{\theta}^{(j)} = -\eta \boldsymbol{v}^{(j)}. \qquad \text{(Equation 25)}$$

Alternatively, in adaptive learning rate methods (Duchi et al., 2011; Kingma and Ba, 2014), the learning rate of individual parameter is adjusted based on the statistics (e.g., mean and variance) of its gradient over training steps. For example, in the Adam method (Kingma and Ba, 2014), the value of a parameter update is magnified if its gradient has been consistent across steps (low variance). Adaptive learning rate methods can be viewed as approximately taking into account curvature of the loss function (Duchi et al., 2011).

Regularization

Regularization techniques are important during training in order to improve generalization performance by deep networks. Adding a L2 regularization term, $L_{\text{reg}} = \lambda \sum_{ij} W_{ij}^2$, to the loss function (Tikhonov, 1943) (equivalent to weight decay; Krogh and Hertz, 1992) discourages the network from using large connection weights, which can improve generalization by implicitly limiting model complexity. Dropout (Srivastava et al., 2014) silences a randomly selected portion of neurons at each step of training. It reduces the network's reliance on particular neurons or a precise combination of neurons. Dropout can be thought of as loosely approximating spiking noise.

The choice of hyperparameters (learning rate, batch size, network initialization, etc.) is often guided by a combination of theory, empirical evidence, and hardware constraints. For neuroscientific applications, it is important that the scientific conclusions do not rely heavily on the hyperparameter choices. And if they do, the dependency should be clearly documented.

3. EXAMPLES OF BUILDING ANNs TO ADDRESS NEUROSCIENCE QUESTIONS

In this section, we overview two common usages of ANNs in addressing neuroscience questions.

3.1. Convolutional Networks for Visual Systems

Deep convolutional neural networks are currently the standard tools in computer vision research and applications (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; He et al., 2016, 2017). These networks routinely consist of tens, sometimes hundreds, of layers of convolutional processing. Effective training of deep feedforward neural networks used to be difficult. This trainability problem has been drastically improved by a combination of innovations in various areas. Modern deep networks would be too large and therefore too slow to run, not to mention train, if not for the rapid development of hardware such as general purpose GPUs (graphics processing units) and TPUs (tensor processing units) (Jouppi et al., 2017). Deep convolutional networks are usually trained with large naturalistic datasets containing millions of high-resolution-labeled images (e.g., Imagenet; Deng et al., 2009), using training methods with adaptive learning rates (Kingma and Ba, 2014; Tieleman and Hinton, 2012). Besides the default use of convolution, a wide range of network architecture innovations improves performance, including the adoption of ReLU activation function (Glorot et al., 2011), normalization methods (Ioffe and Szegedy, 2015), and the use of residual connections that can provide an architectural shortcut from a network layer's inputs directly to its outputs (He et al., 2016).

Deep convolutional networks have been proposed as computational models of the visual systems, particularly of the ventral visual stream or the "what pathway" for visual object information processing (Figure 3) (Yamins and DiCarlo, 2016). These models are typically trained using supervised learning on the same image classification tasks as the ones used in computer vision research and, in many cases, are the exact same convolutional networks developed in computer vision. In comparison, classical models of the visual systems typically rely on hand-designed features (synaptic weights) (Jones and Palmer, 1987; Freeman and Simoncelli, 2011; Riesenhuber and Poggio, 1999), such as Gabor filters, or are trained with unsupervised learning based on the efficient coding principles (Barlow, 1961; Olshausen and Field, 1996). Although classical models have had success at explaining various features of lower-level visual areas, deep convolutional networks surpass them substantially in explaining neural activity in higher-level visual areas in both monkeys (Yamins et al., 2014; Cadieu et al., 2014; Yamins and DiCarlo, 2016) and humans (Khaligh-Razavi and Kriegeskorte, 2014). Besides

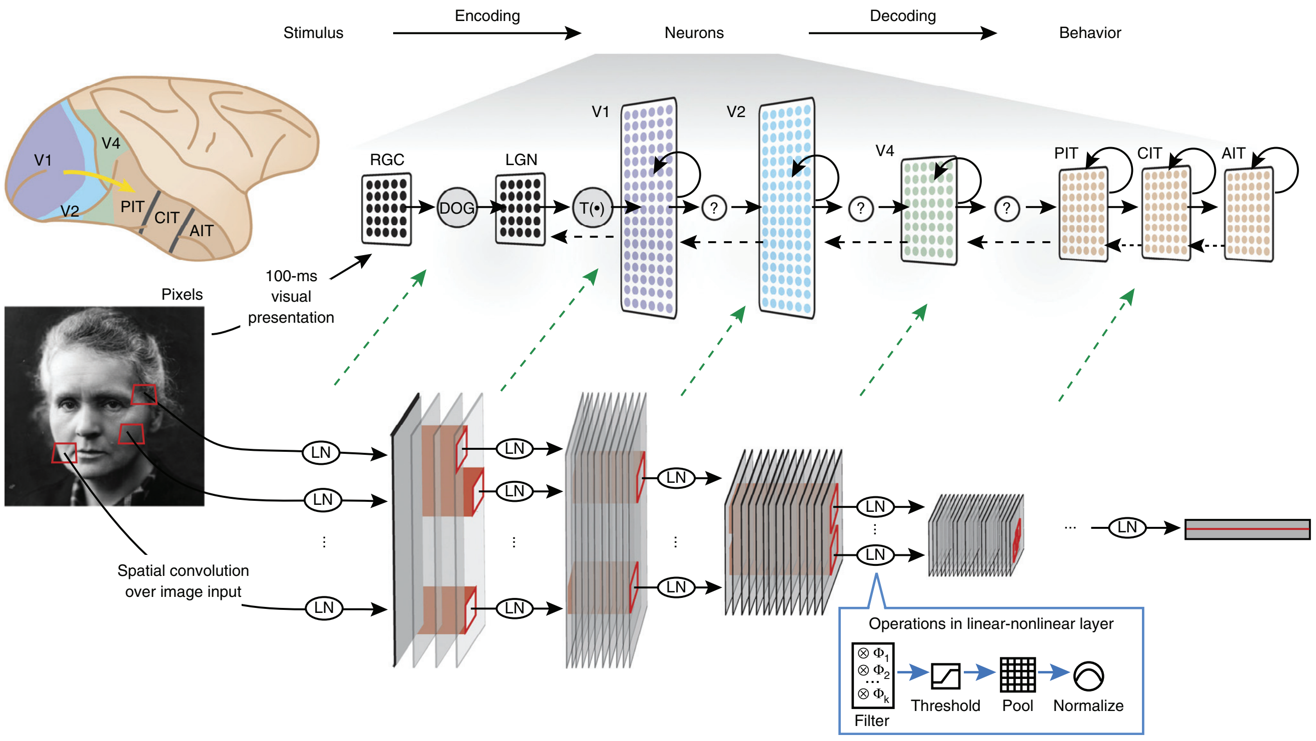

Figure 3. Comparing the Visual System and Deep Convolutional Neural Networks
The same image is passed through monkey's visual cortex (top) and a deep convolutional neural network (bottom), allowing for side-by-side comparisons between biological and ANNs. Neural responses from IT is best predicted by responses from the final layer of the convolutional network, while neural responses from V4 is better predicted by an intermediate network layer (green dashed arrows). Figure adapted from Yamins and DiCarlo (2016).

being trained to classify objects, convolutional networks can also be trained to directly reproduce patterns of neural activity recorded in various visual areas (McIntosh et al., 2016; Prenger et al., 2004).

In a classical work of comparing convolutional networks with higher visual areas (Yamins et al., 2014), Yamins and colleagues trained thousands of convolutional networks with different architectures on a visual categorization task. To study how similar the artificial and biological visual systems are, they quantified how well the network's responses to naturalistic images can be used to linearly predict responses from the inferior temporal (IT) cortex of monkeys viewing the same images. They found that this neural predictivity is highly correlated with accuracy on the categorization task, suggesting that better IT-predicting models can be built by developing better-performing models on challenging natural image classification tasks. They further found that unlike IT, neural responses from the relatively lower visual area, V4, is best predicted by intermediate layers of the networks (Figure 3).

As computational models of visual systems, convolutional networks can model complex, high-dimensional inputs to downstream areas, useful for large-scale models using pixel-based visual inputs (Eliasmith et al., 2012). This process has been made particularly straightforward with the easy access of many pre-trained networks in standard deep learning frameworks like Pytorch (Paszke et al., 2019) and Tensorflow (Abadi et al., 2016).

3.2. RNNs for Cognitive and Motor Systems

RNNs are common machine learning tools to process sequences, such as speech and text. In neuroscience, they have been used to model various aspects of the cognitive, motor, and navigation systems (Mante et al., 2013; Barak et al., 2013; Sussillo et al., 2015; Yang et al., 2019; Wang et al., 2018; Cueva and Wei, 2018). Unlike convolutional networks used to model visual systems that are trained on large-scale image classification tasks, recurrent networks are usually trained on specific cognitive or motor tasks that neuroscientists are studying. By comparing RNNs trained on the same tasks that animals or humans performed, side-by-side comparisons can be made between RNNs and brains. The comparisons can be made at many levels, including single-neuron activity and selectivity, population decoding, state-space dynamics, and network responses to perturbations. We will expand more on how to analyze RNNs in the next section.

An influential work that uses RNNs to model cognition involves a monkey experiment for context-dependent perceptual decision making (Mante et al., 2013). In this task, a fraction (called motion coherence) of random moving dots moves in the same direction (left or right); independently, a fraction (color coherence) of dots are red, and the rest are green. In a single trial, subjects were cued by a context signal to perform either a motion task (judging the net motion direction is right or left) or a color task (deciding whether there are more red dots than green ones). Monkeys performed the task by temporally integrating evidence for behavioral relevant information (e.g., color) while ignoring the irrelevant feature (motion direction in the color task). Neurons in the prefrontal cortex recorded from behaving animals displayed complex activity patterns, where the irrelevant features are still strongly represented even though they weakly influence behavioral choices. These counter-intuitive activity patterns were nevertheless captured by an RNN (Mante et al., 2013). Examining the RNN dynamics revealed a novel mechanism by which the irrelevant features are represented but selectively filtered out and not integrated over time during evidence accumulation.

To better compare neural dynamics between RNNs and biological systems, RNNs used in neuroscience often treat time differently from their counterparts in machine learning. RNNs in machine learning are nearly always discrete time systems (but see Chen et al., 2018), where state at time step t is obtained through a mapping from the state at time step $t-1$ (Equations 14 and 15). The use of a discrete time system means that stimuli that are separated by several seconds in real life can be provided to the network in consecutive time points. To allow for more biologically realistic neural dynamics, RNNs used in neuroscience are often based on continuous time dynamical systems (Wilson and Cowan, 1972; Sompolinsky et al., 1988), such as

$$\tau\frac{d\boldsymbol{r}}{dt} = -\boldsymbol{r}(t) + f(\boldsymbol{W}_r\boldsymbol{r}(t) + \boldsymbol{W}_x\boldsymbol{x}(t) + \boldsymbol{b}_r). \quad \text{(Equation 26)}$$

Here, τ is the single-unit timescale. This continuous-time system can then be discretized using the Euler method with a time step of $\Delta t(<\tau)$,

$$\boldsymbol{r}(t+\Delta t) \approx \boldsymbol{r}(t) + \frac{\Delta t}{\tau}[-\boldsymbol{r}(t) + f(\boldsymbol{W}_r\boldsymbol{r}(t) + \boldsymbol{W}_x\boldsymbol{x}(t) + \boldsymbol{b}_r)]. \quad \text{(Equation 27)}$$

Besides gradient descent through backpropagation, a different line of algorithms has been used to train RNN models in neuroscience (Sussillo and Abbott, 2009; Laje and Buonomano, 2013; Andalman et al., 2019). These algorithms are based on the idea of harnessing chaotic systems with weak perturbations (Jaeger and Haas, 2004). In particular, the FORCE algorithm (Sussillo and Abbott, 2009) allows for rapid learning by modifying the output connections of an RNN to match the target using a recursive least-square algorithm. The network output $y(t)$ (assumed to be one-dimensional here) is fed back to the RNN through $\boldsymbol{w}_{\text{fb}}$,

$$\tau\frac{d\boldsymbol{r}}{dt} = -\boldsymbol{r}(t) + f(\boldsymbol{W}_r\boldsymbol{r}(t) + \boldsymbol{W}_x\boldsymbol{x}(t) + \boldsymbol{w}_{\text{fb}}y(t) + \boldsymbol{b}_r), \quad \text{(Equation 28)}$$

$$y(t) = \boldsymbol{w}_y^{\top}\boldsymbol{r}(t). \quad \text{(Equation 29)}$$

Therefore, modifying the output connections amounts to a low-rank modification $\left(\boldsymbol{w}_{\text{fb}}\boldsymbol{w}_y^{\top}\right)$ of the recurrent connection matrix,

$$\tau\frac{d\boldsymbol{r}}{dt} = -\boldsymbol{r}(t) + f\left(\left[\boldsymbol{W}_r + \boldsymbol{w}_{\text{fb}}\boldsymbol{w}_y^{\top}\right]\boldsymbol{r}(t) + \boldsymbol{W}_x\boldsymbol{x}(t) + \boldsymbol{b}_r\right). \quad \text{(Equation 30)}$$

4. ANALYZING AND UNDERSTANDING ANNs

Common ANNs used in machine learning or neuroscience are not easily interpretable. For many neuroscience problems, they may serve better as model systems that await further analyses. Successful training of an ANN on a task does not mean knowing how the system works. Therefore, unlike most machine learning applications, a trained ANN is not the end goal but merely the prerequisite for analyzing that network to gain understanding.

Most systems neuroscience techniques to investigate biological neural circuits can be directly applied to understand artificial networks. To facilitate side-by-side comparison between artificial and biological neural networks, activity of an ANN can be visualized and analyzed with the same dimensionality reduction tools (e.g., PCA) used for biological recordings (Mante et al., 2013; Kobak et al., 2016; Williams et al., 2018). To understand causal relationship from neurons to behavior, an arbitrary set of neurons can be lesioned (Yang et al., 2019) or inactivated for a short duration, akin to optogenetic manipulation in physiological experiments. Similarly, connections between two selected groups of neurons can be lesioned to understand the causal contribution of cross-population interactions (Andalman et al., 2019).

In this section, we focus on methods that are particularly useful for analyzing ANNs. These methods include optimization-based tuning analysis (Erhan et al., 2009), fixed-point-based dynamical system analysis (Sussillo and Barak, 2013), quantitative comparisons between a model and experimental data (Yamins et al., 2014), and insights from the perspective of biological evolution (Lindsey et al., 2019; Richards et al., 2019).

Similarity Comparison

Analysis methods such as visualization, lesioning, tuning, and fixed-point analysis can offer detailed intuition into the neural mechanisms of individual networks. However, with the relative ease of training ANNs, it is possible to train a large number of neural networks for the same task or dataset (Maheswaranathan et al., 2019; Yamins et al., 2014). With such volume of data, it is necessary to take advantage of high-throughput quantitative methods that compare different models at scale. Similarity comparison methods compute a scalar similarity score between the neural activity of two networks performing the same task (Kriegeskorte et al., 2008; Kornblith et al., 2019). These methods are agnostic about the network form and size and can be applied to artificial and biological networks alike.

Consider two networks (or two populations of neurons), sized N_1 and N_2, respectively. Their neural activity in response to the same D task conditions can be summarized by a D-by-N_1 matrix $\boldsymbol{R}_1$ and a D-by-N_2 matrix $\boldsymbol{R}_2$ (Figure 4A). Representational similarity analysis (RSA) (Kriegeskorte et al., 2008) first computes the dissimilarity or distances of neural responses between different task conditions within each network, yielding a D-by-D dissimilarity matrix for each network (Figure 4B). Next, the correlation between dissimilarity matrices of the two networks is computed. A higher correlation corresponds to more similar representations.

Another related line of methods uses linear regression (as used in Yamins et al., 2014) to predict $\boldsymbol{R}_2$ through a linear

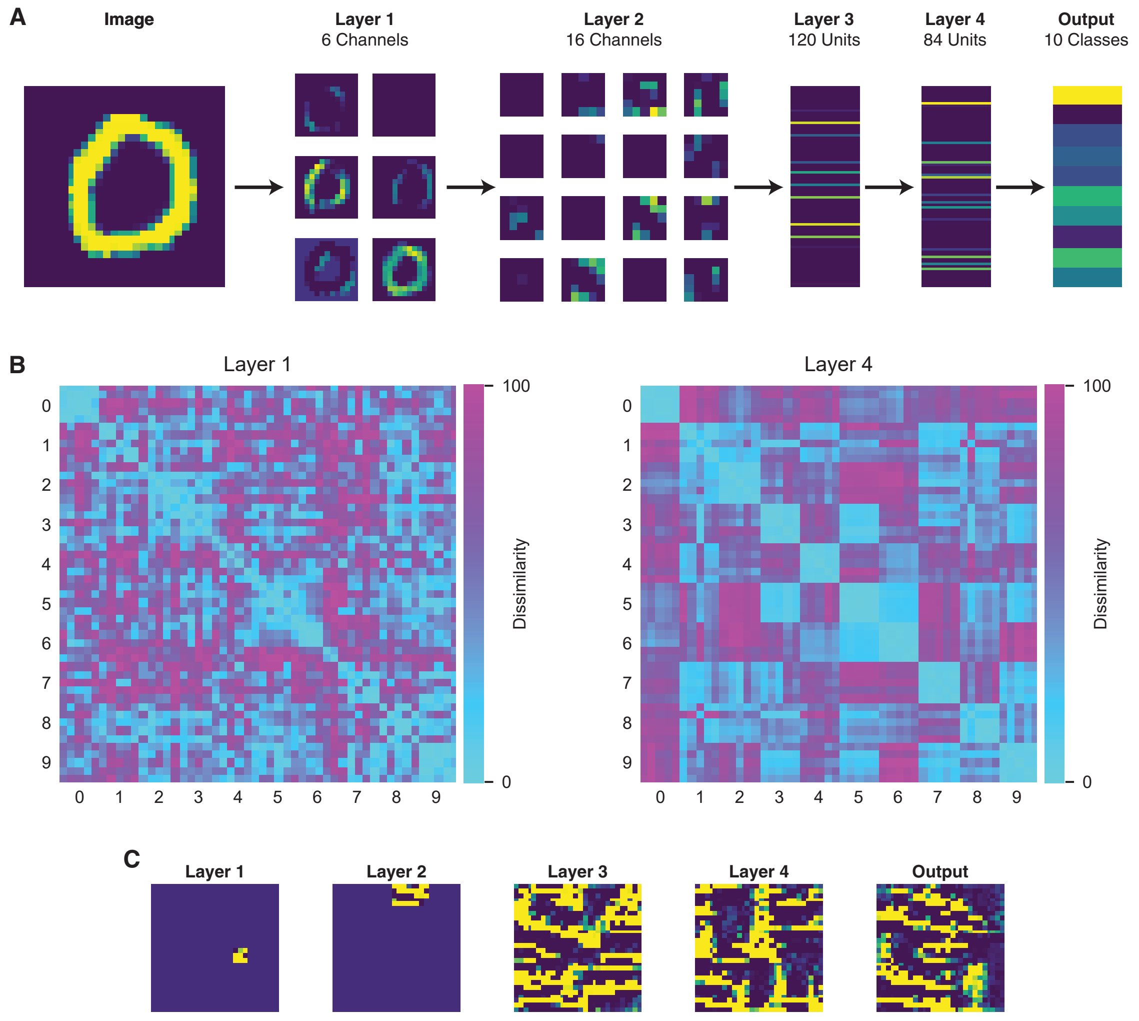

Figure 4. Convolutional Neural Network Responses and Tuning

(A) The neural response to an image in a convolutional neural network trained to classify handwritten digits. The network consists of two layers of convolutional processing, followed by two fully connected layers.

(B) Dissimilarity matrices (each *D*-by-*D*) assessing the similar or dissimilar neural responses to different input images. Dissimilarity matrices are computed for neurons in layers 1 and 4 of the network. $D = 50$. Images are organized by class (0, 1, etc.), five images per class. Neural responses to images in the same class are more similar, i.e., neural representation more category based, in layer 4 (right) than layer 1 (left).

(C) Preferred image stimuli found through gradient-based optimization for sample neurons from each layer. Layers 1 and 2 are convolutional, therefore their neurons have localized preferred stimuli. In contrast, neurons from layers 3 and 4 have non-local preferred stimuli.

transformation of $\boldsymbol{R}_1$, $\boldsymbol{R}_2 \approx \boldsymbol{W}\boldsymbol{R}_1$. The similarity corresponds to the correlation between $\boldsymbol{R}_2$ and its predicted value $\boldsymbol{W}\boldsymbol{R}_1$.

Complex Tuning Analysis

Studying tuning properties of single neurons has been one of the most important analysis techniques in neuroscience (Kuffler, 1953). Classically, tuning properties are studied in sensory areas by showing stimuli parameterized in a low-dimensional space (e.g., oriented bars or gratings in vision; Hubel and Wiesel, 1959). This method is most effective when the neurons studied have relatively simple response properties. A new class of methods treats the mapping of tuning as a high-dimensional optimization problem and directly searches for the stimulus that most strongly activates a neuron. Gradient-free methods such as genetic algorithms have been used to study complex tuning of biological neurons (Yamane et al., 2008). In deep neural networks, gradient-based methods can be used (Erhan et al., 2009; Zeiler and Fergus, 2014). For a neuron with activity $r(\boldsymbol{x})$ given input $\boldsymbol{x}$, a gradient-ascent optimization starts with a random $\boldsymbol{x}_0$ and proceeds by updating the input x as

$$\boldsymbol{x} \rightarrow \boldsymbol{x} + \Delta\boldsymbol{x}; \quad \Delta\boldsymbol{x} = \eta \frac{\partial r}{\partial \boldsymbol{x}}. \qquad \text{(Equation 31)}$$

This method can be used for searching the preferred input to any neuron or any population of neurons in a deep network (Erhan et al., 2009; Bashivan et al., 2019; see Figure 4C for an example). It is particularly useful for studying neurons in higher layers that have more complex tuning properties.

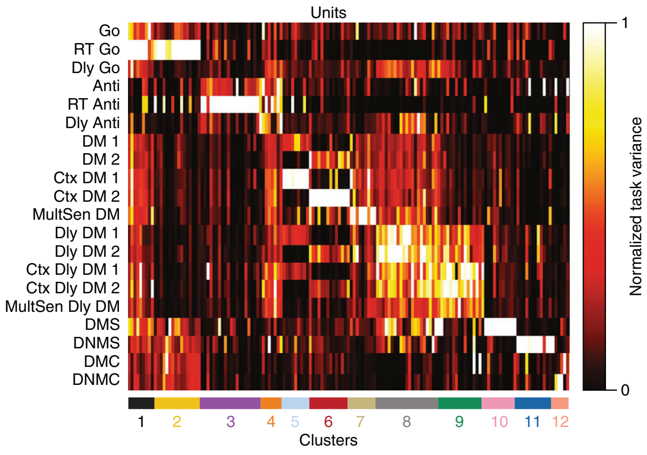

Figure 5. Analyzing Tuning Properties of a Neural Network Trained to Perform 20 Cognitive Tasks

In a network trained on multiple cognitive tasks, the tuning property of model units to individual task can be quantified. x axis, recurrent units; y axis, different tasks. Color measures the degree (between 0 and 1) to which each unit is engaged in a task. Twelve clusters are identified using a hierarchical clustering method (bottom, colored bars). For instance, cluster 3 is highly selective for pro- versus anti-response tasks (Anti) involving inhibitory control; clusters 10 and 11 are involved in delayed match to sample (DMS) and delayed non-match to sample (DNMS), respectively; cluster 12 is tuned to DMC. Figure adapted from Yang et al. (2019).

The space of $\boldsymbol{x}$ may be too high dimensional (e.g., pixel space) for conducting an effective search, especially for gradient-free methods. In that case, we may utilize a lower-dimensional space that is still highly expressive. A generative model learns a function that maps a lower-dimensional latent space to a high-dimensional space such as pixel space (Kingma and Welling, 2013; Goodfellow et al., 2014). Then, the search can be conducted instead in the lower-dimensional latent space (Ponce et al., 2019).

ANNs can be used to build models for complex behavior that would not be easily done otherwise, opening up new possibilities such as studying encoding of a more abstract form of information. For example, Yang et al. (2019) studied neural tuning of task structure, rather than stimuli, in rule-guided problem solving. An ANN was trained to perform many different cognitive tasks commonly used in animal experiments, including perceptual decision making, working memory, inhibitory control, and categorization. Complex network organization is formed by training, in which recurrent neurons display selectivity for a subset of tasks (Figure 5).

Dynamical Systems Analysis

Tuning properties provide a mostly static view of neural representation and computation. To understand how neural networks compute and process information in time, it is useful to study the dynamics of RNNs (Mante et al., 2013; Sussillo and Barak, 2013; Goudar and Buonomano, 2018; Chaisangmongkon et al., 2017).

One useful method to understand dynamics is to study fixed points and network dynamics around them (Strogatz, 2001). In a generic dynamical system,

$$\frac{d\boldsymbol{r}}{dt}=\boldsymbol{F}(\boldsymbol{r}) \qquad \text{(Equation 32)}$$

a fixed point $\boldsymbol{r}_{ss}$ is a steady state where the state does not change in time, $\boldsymbol{F}(\boldsymbol{r}_{ss})=\boldsymbol{0}$. The network dynamics at a state $\boldsymbol{r}=\boldsymbol{r}_{ss}+\Delta\boldsymbol{r}$ around a fixed point $\boldsymbol{r}_{ss}$ is approximately linear,

$$\frac{d\boldsymbol{r}}{dt}=\boldsymbol{F}(\boldsymbol{r})=\boldsymbol{F}(\boldsymbol{r}_{ss}+\Delta\boldsymbol{r})\approx\boldsymbol{F}(\boldsymbol{r}_{ss})+J(\boldsymbol{r}_{ss})\Delta\boldsymbol{r},\ \frac{d\Delta\boldsymbol{r}}{dt}=J(\boldsymbol{r}_{ss})\Delta\boldsymbol{r}. \qquad \text{(Equation 33)}$$

where $\boldsymbol{J}$ is the Jacobian of $\boldsymbol{F}$, $J_{ij}=\partial F_i/\partial r_j$, evaluated at $\boldsymbol{r}_{ss}$. This is a linear system that can be understood more easily, for example, by studying the eigenvectors and eigenvalues of $J(\boldsymbol{r}_{ss})$. In ANNs, these fixed points can be found by gradient-based optimization (Sussillo and Barak, 2013),

$$\text{argmin}_r\|\boldsymbol{F}(\boldsymbol{r})\|^2. \qquad \text{(Equation 34)}$$

Fixed points are particularly useful for understanding how networks store memories, accumulate information (Mante et al., 2013), and transition between discrete states (Chaisangmongkon et al., 2017). This point can be illustrated in a network trained to perform a parametric working memory task (Romo et al., 1999). In this task, a sample vibrotactile stimulus at frequency f_1 is shown, followed by a delay period of a few seconds; then a test stimulus at frequency f_2 is presented, and subjects must decide whether f_2 is higher or lower than f_1 (Figure 6A). During the delay, neurons in the prefrontal cortex of behaving monkeys showed persistent activity at a rate that monotonically varies with f_1. This parametric working memory encoding emerges from training in an RNN (Figure 6B): in the state space of this network, neural trajectories during the delay period converge to different fixed points depending on the stored value. These fixed points form an approximate line attractor (Seung, 1996) during the delay period (Figure 6C).

There is a dearth of examples in computational neuroscience that accounts for not just a single aspect of neural representation or dynamics but a sequence of computation to achieve a complex task. ANNs offer a new tool to confront this difficulty. Chaisangmongkon et al. (2017) used this approach to build a model for delayed match-to-category (DMC) tasks. A DMC task (Figures 6D and 6E) starts with a stimulus sample, say a visual moving pattern, of which a feature (motion direction as an analog quantify from 0° to 360°) is classified into two categories (A in red, B in blue). After a mnemonic delay period, a test stimulus is shown, and the task is to decide whether the test has the same category membership as the sample (Freedman and Assad, 2006). After training to perform this task, a recurrent neural network shows diverse neural activity patterns similar to parietal neurons in monkeys doing the same task (Figure 6F). The trajectory of recurrent neural population in the state space reveals how computation is carried out through epochs of the task (Figure 6G).

Understanding Neural Circuits from Objectives, Architecture, and Training

All above methods seek a mechanistic understanding of ANNs after training. A more integrative view links the three basic ingredients in deep learning: learning problem (tasks/objectives),

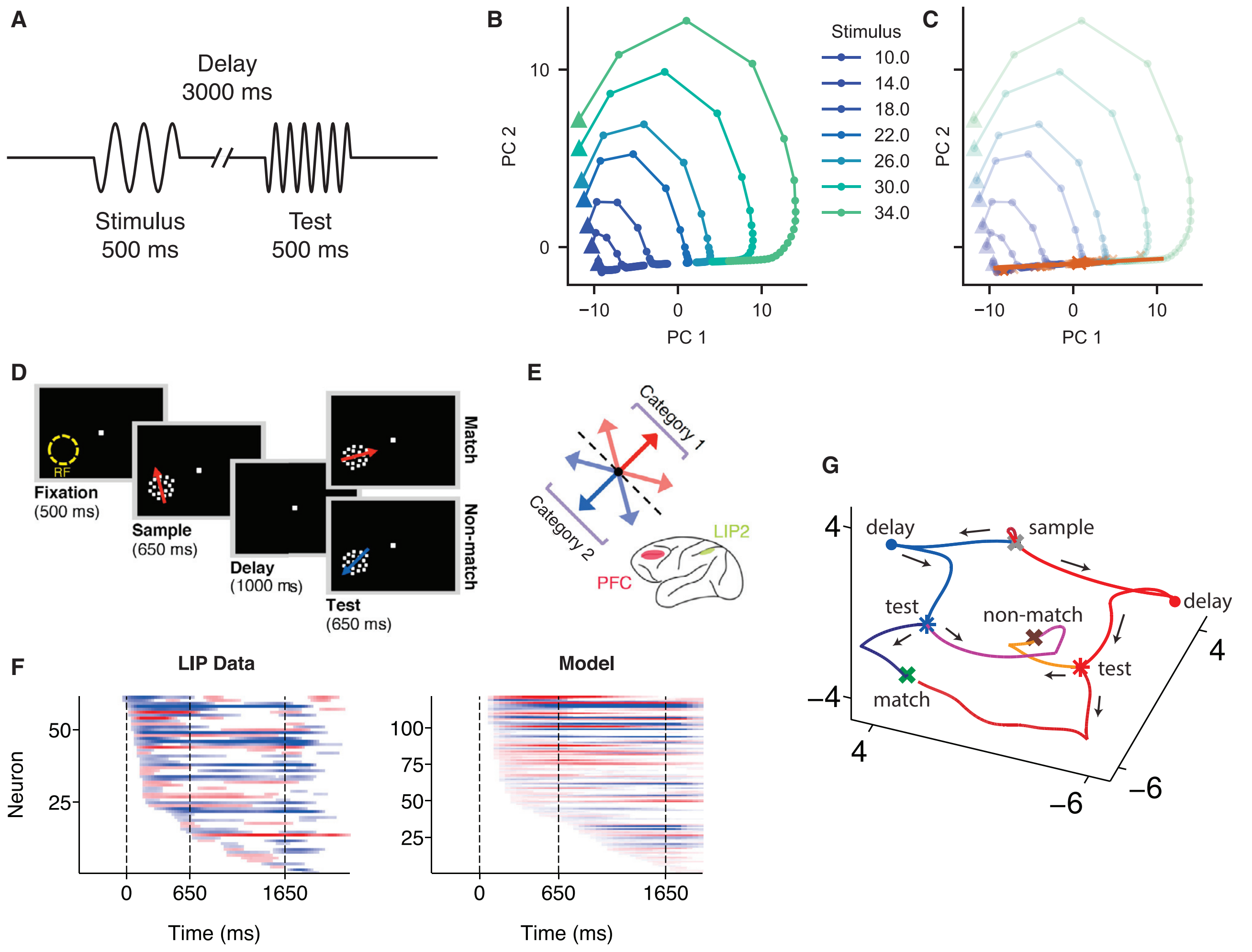

Figure 6. Understanding Network Computation through State Space and Dynamical System Analysis

(A–C) In a simple parametric working memory task (Romo et al., 1999), the network needs to memorize the (frequency) value of a stimulus through a delay period (A). The network can achieve such parametric working memory by developing a line attractor (B and C).

(B) Trial-averaged neural activity during the delay period in the PCA space for different stimulus values. Triangles indicate the start of the delay period.

(C) Fixed points found through optimization (orange cross). The direction of a line attractor can be estimated by finding the eigenvector with a corresponding eigenvalue close to 0. The orange line shows the line attractor estimated around one of the fixed points.

(D–G) Training both recurrent neural networks and monkeys on a delayed match-to-category task (Freedman and Assad, 2006). The task is to decide whether the test and sample stimuli (visual moving pattern) belong to the same category (D). The two categories are defined based on the motion direction of the stimulus (red, category 1; blue, category 2) (E). In an ANN trained to perform this categorization task, the recurrent units of the model display a wide heterogeneity of onset time for category selectivity, similarly to single neurons recorded from monkey posterior parietal cortex (lateral intraparietal area, LIP) during the task (F). Neural dynamics of a recurrent neural network underlying the performance of the DMC task (G). The final decision, match (AA or BB) or non-match (AB or BA) corresponds to distinct attractor states located at separate positions in the state space. Similar trajectories of population activity have been found in experimental data. Figure adapted from Chaisangmongkon et al. (2017).

network architecture, and training algorithm to the solution after training (Richards et al., 2019). This approach is similar to an evolutionary or developmental perspective in biology, which links environments to functions in biological organisms. It can help explain the computational benefit or necessity of observed structures or functions. For example, compared to purely feedforward networks, recurrently connected deep networks are better at predicting responses of higher visual area neurons to behaviorally challenging images of cluttered scenes (Kar et al., 2019). This suggests a contribution of recurrent connections to classifying difficult images in the brain.

While re-running the biological processes of development and evolution may be difficult, re-training networks with different objectives, architectures, and algorithms is fairly straightforward thanks to recent advances in machine learning. Whenever training of an ANN leads to a conclusion, it is good practice to vary hyperparameters describing the basic ingredients (to a reasonable degree) to explore the necessary and sufficient conditions for the conclusion (Orhan and Ma, 2019; Yang et al., 2019; Lindsey et al., 2019).

The link from the three ingredients to the network solution is typically not rigorous. However, in certain simplified cases, the

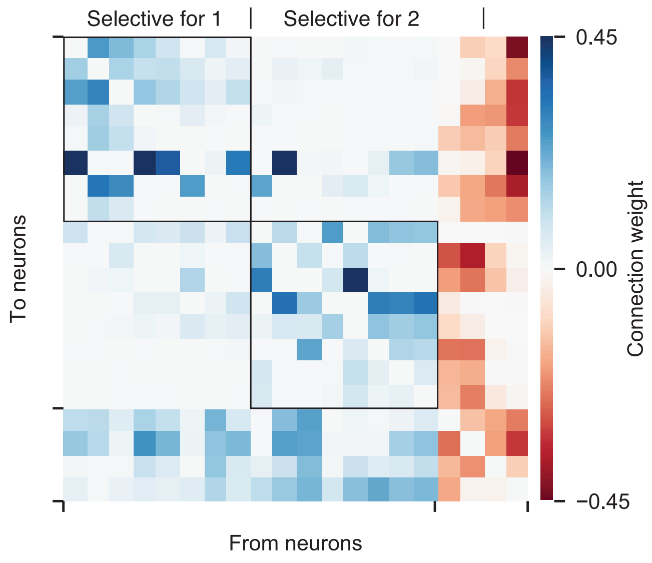

Figure 7. Training a Network with Dale's Law
Connectivity matrix for a recurrent network trained on a perceptual decision-making task. The network respects Dale's law with separate groups of excitatory (blue) and inhibitory (red) neurons. Only connections between neurons with high stimulus selectivity are shown. Neurons are sorted based on their stimulus selectivity to choices 1 and 2. Recurrent excitatory connections between neurons selective to the same choice are indicated by two black squares. Figure inspired by Song et al. (2016).

link can be firmly established by solving the training process analytically (Saxe et al., 2013, 2019b).

5. BIOLOGICALLY REALISTIC NETWORK ARCHITECTURES AND LEARNING

Although neuroscientists and cognitive scientists have had much success with standard neural network architectures (vanilla RNNs) and training algorithms (e.g., SGD) used in machine learning, for many neuroscience questions, it is critical to build network architectures and utilize learning algorithms that are biologically plausible. In this section, we outline methods to build networks with more biologically realistic structures, canonical computations, and plasticity rules.

5.1. Structured Connections

Modern neurophysiological experiments routinely record from multiple brain areas and/or multiple cell types during the same animal behavior. Computational efforts modeling these findings can be greatly facilitated by incorporating into neural networks fundamental biological structures, such as currently known cell-type-specific connectivity and long-range connections across model areas/layers.

In common recurrent networks, the default connectivity is all to all. In contrast, both local and long-range connectivity in biological neural systems are usually sparse. One way to have a sparse connectivity matrix $\boldsymbol{W}$ is by element-wise multiplying a trainable matrix $\tilde{\mathbf{W}}$ with a non-trainable sparse mask $\boldsymbol{M}$, namely $\boldsymbol{W} = \tilde{\boldsymbol{W}} \odot \boldsymbol{M}$. To encourage sparsity without strictly imposing it, a L1 regularization term $\beta \sum_{ij} |W_{ij}|$ can be added to the loss function. The scalar coefficient β controls the strength of the sparsity constraint.

To model cell-type-specific findings, it is important to build neural networks with multiple cell types. A vanilla recurrent network (Equations 14, 15, and 16) (or any other network) can be easily modified to obey Dale's law by separating excitatory and inhibitory neurons (Song et al., 2016),

$$\frac{d\boldsymbol{r}^E}{dt} = -\boldsymbol{r}^E + f_E\left(\boldsymbol{W}_{EE}\boldsymbol{r}^E - \boldsymbol{W}_{EI}\boldsymbol{r}^I + \boldsymbol{W}_{Ex}\boldsymbol{x} + \boldsymbol{b}^E\right), \qquad \text{(Equation 35)}$$

$$\frac{d\boldsymbol{r}^I}{dt} = -\boldsymbol{r}^I + f_I\left(\boldsymbol{W}_{IE}\boldsymbol{r}^E - \boldsymbol{W}_{II}\boldsymbol{r}^I + \boldsymbol{W}_{Ix}\boldsymbol{x} + \boldsymbol{b}^I\right), \qquad \text{(Equation 36)}$$

where an absolute function $|\cdot|$ constrains signs of the connection weights, e.g., $\boldsymbol{W}_{EE} = \left|\tilde{\boldsymbol{W}}_{EE}\right|$. After training an ANN to perform the classical "random dot" task of motion direction discrimination (Roitman and Shadlen, 2002), one can "open the black box" (Sussillo and Barak, 2013) and examine the resulting "wiring diagram" of recurrent network connectivity pattern (Figure 7). With the incorporation of the Dale's law, the connectivity emerging from training is a heterogeneous version of a biologically based structured network model of decision making (Wang, 2002), demonstrating that machine learning brought closer to brain's hardware can indeed be used to shed insights into biological neural networks.

The extensive long-range connectivity across brain areas (Felleman and Van Essen, 1991; Markov et al., 2014; Oh et al., 2014) can be included in ANNs. In classical convolutional neural networks (LeCun et al., 1990; Krizhevsky et al., 2012), each layer only receives feedforward inputs from the immediate preceding layer. However, in some recent networks, each layer also receives feedforward inputs from much earlier layers (Huang et al., 2017; He et al., 2016). In convolutional recurrent networks, neurons in each layer further receive feedback inputs from later layers and local recurrent connections (Nayebi et al., 2018; Kietzmann et al., 2019).

5.2. Canonical Computation

Neuroscientists have identified several canonical computations that are carried out across a wide range of brain areas, including attention, normalization, and gating. Here, we discuss how such canonical computations can be introduced into neural networks. They function as modular architectural components that can be plugged into many networks. Interestingly, canonical computations mentioned above all have their parallels in machine-learning-based neural networks. We will highlight the differences and similarities between purely machine learning implementations and more biological ones.

Normalization

Divisive normalization is widely observed in biological neural systems (Carandini and Heeger, 2011). In divisive normalization, activation of a neuron r_i is no longer determined by its immediate input I_i, $r_i = f(I_i)$. Instead, it is normalized by the sum of inputs $\sum_j I_j$ to a broader pool of neurons called the normalization pool,

$$r_i = f\left(\gamma \frac{I_i}{\sum_j I_j + \sigma}\right). \qquad \text{(Equation 37)}$$

The specific choice of a normalization pool depends on the system studied. Biologically, although synaptic inputs are additive in the drive to neurons, feedback inhibition can effectively produce normalization (Ardid et al., 2007). This form of divisive normalization is differentiable. So, it can be directly incorporated into ANNs.

Normalization is also a critical part of many neural networks in machine learning. Similar to divisive normalization, machine-learning-based normalization methods (Ioffe and Szegedy, 2015; Ba et al., 2016b; Ulyanov et al., 2016; Wu and He, 2018) aim at putting neuronal responses into a range appropriate for downstream areas to process. Unlike divisive normalization, the mean inputs to a pool of neurons is usually subtracted from, instead of dividing, the immediate input (Equation 21). These methods also compute the standard deviation of inputs to the normalization pool, a step that may not be biologically plausible. Different machine-learning-based normalization methods are distinguished based on their choice of a normalization pool.

Attention

Attention has been extensively studied in neuroscience (Desimone and Duncan, 1995; Carrasco, 2011). Computational models are able to capture various aspects of bottom-up (Koch and Ullman, 1987) and top-down attention (Reynolds and Heeger, 2009). In computational models, top-down attention usually takes the form of a multiplicative gain field to the activity of a specific group of neurons. In the case of spatial attention, consider a group of neurons, each with a preferred spatial location x_i and pre-attention activity $\tilde{r}(x_i)$ for a certain stimulus. The attended spatial location x_q results in attentional weights $\alpha_i(x_q)$, which is higher if x_q is similar to x_i. The attentional weights can then be used to modulate the neural response of neuron i, $r_i(x_q) = \alpha_i(x_q)\tilde{r}(x_i)$. Similarly, feature attention strengthens the activity of neurons that are selective to the attended features (e.g., specific color). Such top-down spatial and feature attention can be included in convolutional neural networks (Lindsay and Miller, 2018; Yang et al., 2018).

Meanwhile, attention has become widely used in machine learning (Bahdanau et al., 2016; Xu et al., 2015; Lindsay, 2020), constituting a standard component in recent natural language processing models (Vaswani et al., 2017). Although the machine learning attention mechanisms appear rather different from attention models in neuroscience, as we will show below, the two mechanisms are very closely related.

In deep learning, attention can be viewed as a differentiable dictionary retrieval process. A regular dictionary stores a number of key-value pairs (e.g., word-explanation pairs) $\left\{\left(\boldsymbol{k}^{(i)}, \boldsymbol{v}^{(i)}\right)\right\}$, similar to looking up an explanation $\left(\boldsymbol{v}^{(i)}\right)$ of a word $\left(\boldsymbol{k}^{(i)}\right)$. For a given query $\boldsymbol{q}$, using a dictionary involves searching for the key $\boldsymbol{k}^{(j)}$ that matches $\boldsymbol{q}$, $\boldsymbol{k}^{(j)} = \boldsymbol{q}$, and retrieving the corresponding value, $\boldsymbol{y} = \boldsymbol{v}^{(j)}$. This process can be thought of as modulating each value $\boldsymbol{v}^{(i)}$ based on an attentional weight α_i that measures the similarity between the key $\boldsymbol{k}^{(i)}$ and the query $\boldsymbol{q}$. In the simple binary case,

$$\alpha_i = \begin{cases} 1, & \text{if } \boldsymbol{k}^{(i)} = \boldsymbol{q} \\ 0, & \text{otherwise} \end{cases} \qquad \text{(Equation 38)}$$

which modulated the output as

$$\boldsymbol{y} = \sum_i \alpha_i \boldsymbol{v}^{(i)}. \qquad \text{(Equation 39)}$$

In the above case of spatial attention, the i-th key-value pair is $(x_i, \tilde{r}(x_i))$, while the query is the attended spatial location x_q. Each neuron's response is modulated based on how similar its preferred spatial location (its value) x_i is to the attended location (the query) x_q.

The use of machine learning attention makes the query-key comparison and the value-retrieval process differentiable. A query is compared with every key vector $\boldsymbol{k}^{(i)}$ to obtain an attentional weight (normalized similarity score) α_i,

$$c_i = \text{score}\left(\boldsymbol{q}, \boldsymbol{k}^{(i)}\right), \qquad \text{(Equation 40)}$$

$$\alpha_1, \cdots, \alpha_N = \text{normalize}(c_1, \cdots, c_N), \qquad \text{(Equation 41)}$$

Here, the similarity scoring function can be a simple inner product, $\text{score}\left(\boldsymbol{q}, \boldsymbol{k}^{(i)}\right) = \boldsymbol{q}^{\top}\boldsymbol{k}^{(i)}$ (Bahdanau et al., 2016), and the normalization function can be the softmax function,

$$\alpha_i = \frac{e^{c_i}}{\sum_j e^{c_j}}, \quad \text{such that} \quad \sum_i \alpha_i = 1. \qquad \text{(Equation 42)}$$

The use of a normalization function is critical, as it effectively forces the network to focus on a few key vectors (a few attended locations in the case of spatial attention).

Gating

An important computation for biological neural systems is gating (Abbott, 2006; Wang and Yang, 2018). Gating refers to the idea of controlling information flow without necessarily distorting its content. Gating in biological systems can be implemented with various mechanisms. Attention modulation multiplies inputs to neurons by a gain factor, providing a graded mechanism of gating at the level of sensory systems (Salinas and Thier, 2000; Olsen et al., 2012). Another form of gating may involve several types of inhibitory neurons (Wang et al., 2004; Yang et al., 2016). At the behavioral level, gating often appears to be all or none, as exemplified by effects such as inattentional blindness.

In deep learning, multiplicative gating is essential for popular recurrent network architectures such as LSTM (long short-term-memory) networks (Equation 43) (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) and GRU (gated recurrent unit) networks (Cho et al., 2014; Chung et al., 2014). Gated networks are generally easier to train and more powerful than vanilla

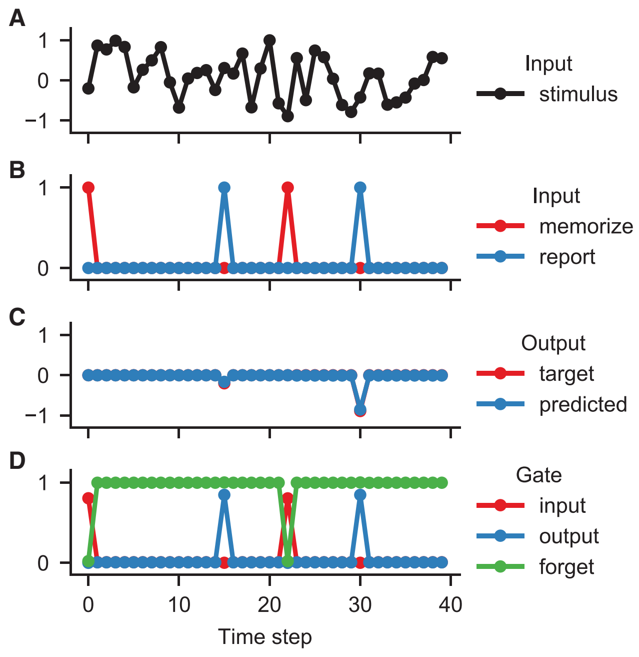

Figure 8. Visualizing LSTM Activity in a Simple Memory Task
(A–C) A simple memory task.
(A) The network receives a stream of input stimulus, the value of which is randomly and independently sampled at each time point.
(B) When the "memorize input" (red) is active, the network needs to remember the current value of the stimulus (A) and output that value when the "report input" (blue) is next active.
(C) After training, a single-unit LSTM can perform the task almost perfectly for modest memory duration.
(D) When the memorize input is active, this network opens the input gate (allowing inputs) and closes the forget gate (forgetting previous memory). It opens the output gate when the report input is active.

RNNs. Gating variables dynamically control information flow within these networks through multiplicative interactions. In a LSTM network, there are three types of gating variables. Input and output gates, $\boldsymbol{g}_t^i$ and $\boldsymbol{g}_t^o$, control the inputs to and outputs of the cell state $\boldsymbol{c}_t$, while forget gate $\boldsymbol{g}_t^f$ controls whether cell state $\boldsymbol{c}_t$ keeps its memory $\boldsymbol{c}_{t-1}$.

$$\boldsymbol{g}_t^f = \sigma_g(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\boldsymbol{r}_{t-1} + \boldsymbol{b}_f), \qquad \text{(Equation 43)}$$

$$\boldsymbol{g}_t^i = \sigma_g(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\boldsymbol{r}_{t-1} + \boldsymbol{b}_i),$$

$$\boldsymbol{g}_t^o = \sigma_g(\boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{U}_o\boldsymbol{r}_{t-1} + \boldsymbol{b}_o),$$

$$\boldsymbol{c}_t = \boldsymbol{g}_t^f \odot \boldsymbol{c}_{t-1} + \boldsymbol{g}_t^i \odot \sigma_c(\boldsymbol{W}_c\boldsymbol{x}_t + \boldsymbol{U}_c\boldsymbol{r}_{t-1} + \boldsymbol{b}_c),$$

$$\boldsymbol{r}_t = \boldsymbol{g}_t^o \odot \sigma_r(\boldsymbol{c}_t).$$

Here, the symbol $\odot$ denotes the element-wise (Hadamard) product of two vectors of the same length ($\boldsymbol{z} = \boldsymbol{x} \odot \boldsymbol{y}$ means $z_i = x_i y_i$). Gating variables are bounded between 0 and 1 by the sigmoid function σ_g, which can be viewed as a smooth differentiable approximate of a binary step function. A gate is opened or closed when its corresponding gate value is near 1 or 0, respectively. All the weights ($\boldsymbol{W}$ and $\boldsymbol{U}$ matrices) are trained. By introducing these gates, a LSTM can, in principle, keep a memory in its cell state $\boldsymbol{c}_t$ indefinitely by having the forget gate $\boldsymbol{g}_t^f = \boldsymbol{1}$ and input gate $\boldsymbol{g}_t^i = \boldsymbol{0}$ (Figure 8). In addition, the network can choose when to read out from the memory by setting its output gate $\boldsymbol{g}_t^o = \boldsymbol{0}$ or $\boldsymbol{1}$. Despite their great utility to machine learning, LSTMs (and GRUs) cannot be easily related to biological neural circuits. Modifications to LSTMs have been suggested so the gating process could be better explained by neurobiology (Costa et al., 2017).

Although both attention and gating utilize multiplicative interactions, a critical difference is that in attention, the neural modulation is normalized (Equation 42), whereas in gating it is not. Therefore, neural attention often has one focus, while neural gating can open or close gates to all neurons uniformly. An important insight from machine learning is that gating should be plastic, which should inspire neuroscientists to investigate learning to gate in the brain.

Predictive Coding

Another canonical computation proposed for the brain is to compute predictions (Rao and Ballard, 1999; Bastos et al., 2012; Heilbron and Chait, 2018). In predictive coding, a neural system constantly tries to make inference about the external world. Brain areas will selectively propagate information that is unpredicted or surprising while suppressing responses to expected stimuli. To implement predictive coding in ANNs, feedback connections from higher layers can be trained with a separate loss that compares the output of feedback connections with the neural activity in lower layers (Lotter et al., 2016; Sacramento et al., 2018). In this way, feedback connections will learn to predict the activity of lower areas. The feedback inputs will then be used to inhibit neural activity in lower layers.

5.3. Learning and Plasticity

Biological neural systems are products of evolution, development, and learning. In contrast, traditional ANNs are trained with SGD-based rules mostly from scratch. The backpropagation algorithm of computing gradient descent is well known to be biologically implausible (Zipser and Andersen, 1988). Incorporating more realistic learning processes can help us build better models of brains.

Selective Training and Continual Learning

In typical ANNs, all connections are trained. However, in biological neural systems, synapses are not equally modifiable. Many synapses can be stable for years (Grutzendler et al., 2002; Yang et al., 2009). To implement selective training of connections, the effective connection matrix $\boldsymbol{W}$ can be expressed as a sum of a sparse trainable synaptic weight matrix and a non-trainable one, $\boldsymbol{W} = \boldsymbol{W}_{\text{train}} + \boldsymbol{W}_{\text{fix}}$ (Rajan et al., 2016; Masse et al., 2018). Or more generally, selective training can be imposed softly by adding to the loss a regularization term L_{reg} that makes it more difficult to change the weights of certain connections,

$$L_{\text{reg}} = \beta \sum_{ij} M_{ij}(W_{ij} - W_{\text{fix},ij})^2. \qquad \text{(Equation 44)}$$

Here, M_{ij} determine how strongly the connection W_{ij} should stick close to the value $W_{\mathrm{fix},ij}$.

Selective training of connections through this form of soft constraints has been used by continual learning techniques to combat catastrophic forgetting. The phenomenon of catastrophic forgetting is commonly observed when ANNs are learning new tasks; they tend to rapidly forget previous learned tasks that are not revisited (McCloskey and Cohen, 1989). One major class of continual learning methods deals with this issue by selectively training synaptic connections that are deemed unimportant for previously learned tasks or knowledge while protecting the important ones (Kirkpatrick et al., 2017; Zenke et al., 2017).

Hebbian Plasticity

The predominant idea for biological learning is Hebbian plasticity (Hebb, 2005) and its variants (Song et al., 2000; Bi and Poo, 2001). Hebbian plasticity is an unsupervised learning method that drives learning of connection weights without target outputs or rewards. It is essential for classical models of associative memory, such as Hopfield networks (Hopfield, 1982), and has a deep link to modern neural network architectures with explicit long-term memory modules (Graves et al., 2014).

Supervised learning techniques, especially those based on SGD, can be combined with Hebbian plasticity to develop ANNs that are both more powerful for certain tasks and more biologically realistic. There are two methods to combine Hebbian plasticity with SGD. In the first kind, the effective connection matrix $\boldsymbol{W} = \tilde{\boldsymbol{W}} + \boldsymbol{A}$ is the sum of two connection matrices, $\tilde{\boldsymbol{W}}$ trained by SGD, and $\boldsymbol{A}$ driven by Hebbian plasticity (Ba et al., 2016a; Miconi et al., 2018),

$$\boldsymbol{A}(t+1) = \lambda \boldsymbol{A}(t) + \eta \boldsymbol{r}\boldsymbol{r}^{\top}. \qquad \text{(Equation 45)}$$

Or in component form,

$$A_{ij}(t+1) = \lambda A_{ij}(t) + \eta r_i r_j. \qquad \text{(Equation 46)}$$

In addition to training a separate matrix, SGD can be used to learn the plasticity rules itself (Bengio et al., 1992; Metz et al., 2018). Here, the plasticity rule is a trainable function of pre- and postsynaptic activity,

$$A_{ij}(t+1) = \lambda A_{ij}(t) + f(r_i, r_j, \boldsymbol{\theta}). \qquad \text{(Equation 47)}$$

Because the system is differentiable, parameters $\boldsymbol{\theta}$, which collectively describe the plasticity rules, can be updated with SGD-based methods. In its simplest form, $f(r_i, r_j, \boldsymbol{\theta}) = \eta r_i r_j$, where $\boldsymbol{\theta} = \{\eta\}$. Here, the system can learn to become Hebbian ($\eta > 0$) or anti-Hebbian ($\eta < 0$). Learning of a plasticity rule is a form of meta-learning, using an algorithm (here, SGD) to optimize an inner learning rule (here, Hebbian plasticity).

Such Hebbian plasticity networks can be extended to include more complex synapses with multiple hidden variables in a "cascade model" of synaptic plasticity (Fusi et al., 2005). In theory, properly designed complex synapses can substantially boost a neural network's memory capacity (Benna and Fusi, 2016). Models of such complex synapses are differentiable and therefore can be incorporated into ANNs (Kaplanis et al., 2018).

Short-Term Plasticity

In addition to Hebbian plasticity that acts on the timescales from hours to years, biological synapses are subject to short-term plasticity mechanisms operating on the timescale of hundreds of milliseconds to seconds (Zucker and Regehr, 2002) that can rapidly modify their effective weights. Classical short-term plasticity rules (Mongillo et al., 2008; Markram et al., 1998) are formulated with spiking neurons, but they can be adapted to rate forms. In these rules, each connection weight $w = \tilde{w}ux$ is a product of an original weight $\tilde{w}$, a facilitating factor u, and a depressing factor x. The facilitating and depressing factors are both influenced by the presynaptic activity $r(t)$,

$$\frac{dx}{dt} = \frac{1 - x(t)}{\tau_x} - u(t)x(t)r(t), \qquad \text{(Equation 48)}$$

$$\frac{du}{dt} = \frac{U - u(t)}{\tau_u} + U(1 - u(t))r(t). \qquad \text{(Equation 49)}$$

High presynaptic activity $r(t)$ increases the facilitating factor $u(t)$ and decreases the depressing factor $x(t)$. Again, the equations governing short-term plasticity are fully differentiable, so they can be incorporated into ANNs in the same way as Hebbian plasticity rules (Masse et al., 2019).

Masse et al. (2019) offers an illustration of how ANNs can be used to test new hypotheses in neuroscience. It was designed to investigate the neural mechanisms of working memory, the brain's ability to maintain and manipulate information internally in the absence of external stimulation. Working memory has been extensively studied in animal experiments using delayed response tasks, in which a stimulus and its corresponding motor response are separated by a temporal gap when the stimulus must be retained internally. Stimulus-selective self-sustained persistent activity during a mnemonic delay is amply documented and considered as the neural substrate of working memory representation (Goldman-Rakic, 1995; Wang, 2001). However, recent studies suggested that certain short-term memory traces may be realized by hidden variables instead of spiking activity, such as synaptic efficacy that by virtue of short-term plasticity represents past events (Stokes, 2015; Mongillo et al., 2008). When an ANN endowed with short-term synaptic plasticity is trained to perform a delayed response task, it does not make an *a priori* assumption about whether working memory is represented by hidden synaptic efficacy or neural activity. It was found that activity-silent state can accomplish such a task only when the delay is sufficiently short, whereas persistent activity naturally emerges from training with delay periods longer than the biophysical time constants of short-term synaptic plasticity. More importantly, training always gives rise to persistent activity, even with a short mnemonic delay period, when information must be manipulated internally, such as mentally rotating a directional stimulus by 90°. This work illustrates how ANNs can contribute to resolving important debates in neuroscience.

Biologically Realistic Gradient Descent

Backpropagation is commonly viewed as biologically unrealistic because the plasticity rule is not local (see Equation 13). Efforts have been devoted to approximating gradient descent with algorithms more compatible with the brain's hardware (Lillicrap et al., 2016; Guerguiev et al., 2017; Roelfsema and Holtmaat, 2018; Lillicrap et al., 2020).

In feedforward networks, the backpropagation algorithm can be implemented with synaptic connections feeding back from the final layer (Xie and Seung, 2003). This implementation assumes that the feedback connections precisely mirror the feedforward connections. This requirement can be relaxed. If a network uses fixed and random feedback connections, the feedforward connections would start to approximately mirror the feedback connections during training (a phenomenon called "feedback alignment"), allowing for training loss to be decreased (Lillicrap et al., 2016). Another challenge of approximating backpropagation with feedback connections is that the feedback inputs carrying loss information need to be processed differently from feedforward inputs carrying stimulus information. This issue can be addressed by introducing multi-compartmental neurons into ANNs (Guerguiev et al., 2017). In such networks, feedforward and feedback inputs are processed separately because they are received by the model neurons' soma and dendrites, respectively.

These methods of implementing the backpropagation algorithm through synapses propagating information backward are so far only used for feedforward networks. For recurrent networks, the backpropagation algorithm propagates information backward in time. Therefore, it is not clear how to interpret the backpropagation in terms of synaptic connections. Instead, approximations can be made such that the network computes approximated gradient information as it runs forward in time (Williams and Zipser, 1989; Murray, 2019).

For many neuroscientific applications, it is probably not necessary to justify backpropagation by neurobiology. ANNs often start as "blank slate"; thus, training by backpropagation is tasked to accomplish what for the brain amounts to a combination of genetic programming, development, and plasticity in adulthood.

6. FUTURE DIRECTIONS AND CONCLUSION

Recent years have seen a growing impact of ANN models in neuroscience. We have reviewed many of these efforts in the section "Biologically Realistic Network Architectures and Learning." In this final section, we outline other existing challenges and ongoing work to make ANNs better models of brains.

Spiking Neural Networks

Most biological neurons communicate with spikes. Harnessing the power of machine learning algorithms for spiking networks remains a daunting challenge. Gradient-descent-based training techniques typically require the system to be differentiable, making it challenging to train spiking networks because spike generation is non-differentiable. However, several recent methods have been proposed to train spiking networks with gradient-based techniques (Courbariaux et al., 2016; Bellec et al., 2018; Zenke and Ganguli, 2018; Nicola and Clopath, 2017; Huh and Sejnowski, 2018). These methods generally involve approximating spike generation with a differentiable system during backpropagation (Tavanaei et al., 2019). Techniques to effectively train spiking networks could prove increasingly important and practical, as neuromorphic hardware that operates naturally with spikes becomes more powerful (Merolla et al., 2014; Pei et al., 2019).

Standardized Protocols for Developing Brain-like Recurrent Networks

In the study of mammalian visual systems, the use of large datasets such as ImageNet (Deng et al., 2009) was crucial for producing neural networks that resemble biological neural circuits in the brain. The same has not been shown for most other systems. Although many studies have shown success using neural networks to model cognitive and motor systems, each work usually has its own set of network architectures, training protocols, and other hyperparameters. Simply applying the most common architectures and training algorithms does not consistently lead to brain-like recurrent networks (Sussillo et al., 2015). Much work remains to be done to search for datasets/tasks, network architectures, and training regimes that can produce brain-resembling artificial networks across a wide range of experimental tasks.

Detailed Behavioral and Physiological Predictions

Although many studies have reported similarities between brains and ANNs, more detailed comparisons have revealed striking differences (Szegedy et al., 2013; Hénaff et al., 2019; Sussillo et al., 2015). Deep convolutional networks can achieve similar or better performance on large image classification tasks compared to humans; however, the mistakes they make can be very different from the ones made by humans (Szegedy et al., 2013; Rajalingham et al., 2018). It will be important for future ANN models of brains to aim at simultaneously explaining a wider range of physiological and behavioral phenomena.

Interpreting Learned Networks and Learning Processes

With the ease of training neural networks comes the difficulty of analyzing them. Granted, neuroscientists are not foreign to analysis of complex networks, and ANNs are still technologically easier to analyze compared to biological neural networks. However, compared to network models with built-in regularities and small numbers of free parameters, deep neural networks are notoriously complex to analyze and understand and will likely become even more so as we build more and more sophisticated neural networks. This difficulty is rooted in the use of optimization algorithms to search for parameter values. Since the optimization process in deep learning has no unique optima, the results of optimization necessarily lack the degree of regularities built in hand-designed models. Although we can attempt to understand ANNs from the perspective of its objectives, architectures, and training algorithms (Richards et al., 2019), which are described with a much smaller number of hyperparameters, the link from these hyperparameters to network representation, mechanism, and behavior is mostly informal and based on intuition.

Despite the difficulties mentioned above, several lines of research hold promise. To facilitate understanding of learned networks, one can construct variants of neural networks that are more interpretable. For example, low-rank recurrent neural networks utilize recurrent connectivity matrices with low-dimensional structures (Mastrogiuseppe and Ostojic, 2018), allowing for a more straightforward mapping from network connectivity to dynamics and computation.

The dynamics of learning in neural networks can be studied analytically in deep linear networks (Saxe et al., 2013) and very wide nonlinear networks, i.e., networks with a sufficiently large number of neurons per layer (Jacot et al., 2018). In another line of work, the Information Bottleneck theory proposes that learning processes in neural networks are characterized by two phases: the first extracts information for output tasks (prediction), and the second discards (excessive) information about inputs (compression) (Shwartz-Ziv and Tishby, 2017; see also Saxe et al., 2019a). Progress in these directions could shed light on why neural networks can generalize to new data despite having many parameters, which would traditionally indicate overfitting and poor generalization performance.

Conclusion

ANNs present a novel approach in computational neuroscience. They have already been used, with a certain degree of success, to model various aspects of sensory, cognitive, and motor circuits. Efforts are underway to make ANNs more biologically relevant and applicable to a wider range of neuroscientific questions. In a sense, instead of being viewed as computational models, ANNs can be studied as model systems, like fruit flies, mice, and monkeys, but are easily carried out to explore new task paradigms and computational ideas. Of course, one can be skeptical about ANNs as model systems, on the ground that they are not biological organisms. However, computational models span a wide range of biological realism; there should be no doubt that brain research will benefit from enhanced interactions with machine learning and artificial intelligence. In order for ANNs to have a broad impact in neuroscience, it will be important to devote our efforts in two areas. First, we should continue to bring ANNs closer to neurobiology. Second, we should endeavor to "open the black box" thoroughly after learning to identify neural representation, temporal dynamics, and network connectivity that emerge from learning, leading to testable insights and predictions by neurobiological experiments. Recurrent neural dynamics emphasized in this Primer represent a salient feature of the brain; further development of strongly recurrent ANNs will contribute to acceleration of progress in neuroscience.

ACKNOWLEDGMENTS

We thank Vishwa Goudar and Jacob Portes for helpful comments on a draft of this paper. This work was supported by the Simons Foundation, NSF NeuroNex Award DBI-1707398, and the Gatsby Charitable Foundation to G.R.Y. and the ONR grant N00014 and Simons Collaboration in the Global Brain (SCGB) (grant 543057SPI) to X.-J.W.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 265–283.

Abbott, L. (2006). Where are the switches on this thing? In 23 Problems in Systems Neuroscience, J.L. van Hemmen and T.J. Sejnowski, eds. (Oxford University Press), pp. 423–431.

Abbott, L.F. (2008). Theoretical neuroscience rising. Neuron *60*, 489–495.

Abbott, L.F., and Chance, F.S. (2005). Drivers and modulators from push-pull and balanced synaptic input. Prog. Brain Res. *149*, 147–155.

Andalman, A.S., Burns, V.M., Lovett-Barron, M., Broxton, M., Poole, B., Yang, S.J., Grosenick, L., Lerner, T.N., Chen, R., Benster, T., et al. (2019). Neuronal dynamics regulating brain and behavioral state transitions. Cell *177*, 970–985.e20.

Ardid, S., Wang, X.-J., and Compte, A. (2007). An integrated microcircuit model of attentional processing in the neocortex. J. Neurosci. *27*, 8486–8495.

Ba, J., Hinton, G.E., Mnih, V., Leibo, J.Z., and Ionescu, C. (2016a). Using fast weights to attend to the recent past. Adv. Neural Inf. Process. Syst. *29*, 4331–4339.

Ba, J.L., Kiros, J.R., and Hinton, G.E. (2016b). Layer normalization. arXiv, 1607.06450 https://arxiv.org/abs/1607.06450.

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv. https://arxiv.org/abs/1409.0473.

Barak, O. (2017). Recurrent neural networks as versatile tools of neuroscience research. Curr. Opin. Neurobiol. *46*, 1–6.

Barak, O., Sussillo, D., Romo, R., Tsodyks, M., and Abbott, L.F. (2013). From fixed points to chaos: three models of delayed discrimination. Prog. Neurobiol. *103*, 214–222.

Barlow, H.B. (1961). Possible principles underlying the transformation of sensory messages. Sensory Communication *1*, 217–234.

Bashivan, P., Kar, K., and DiCarlo, J.J. (2019). Neural population control via deep image synthesis. Science *364*, eaav9436.

Bastos, A.M., Usrey, W.M., Adams, R.A., Mangun, G.R., Fries, P., and Friston, K.J. (2012). Canonical microcircuits for predictive coding. Neuron *76*, 695–711.

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. (2018). Long short-term memory and learning-to-learn in networks of spiking neurons. Adv. Neural Inf. Process. Syst. *31*, 787–797.

Bengio, S., Bengio, Y., Cloutier, J., and Gecsei, J. (1992). On the optimization of a synaptic learning rule. In Preprints Conf. Optimality in Artificial and Biological Neural Networks (University of Texas).

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. *5*, 157–166.

Benna, M.K., and Fusi, S. (2016). Computational principles of synaptic memory consolidation. Nat. Neurosci. *19*, 1697–1706.

Bi, G., and Poo, M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. Annu. Rev. Neurosci. *24*, 139–166.

Bottou, L., Curtis, F.E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. SIAM Rev. *60*, 223–311.

Botvinick, M., Wang, J.X., Dabney, W., Miller, K.J., and Kurth-Nelson, Z. (2020). Deep reinforcement learning and its neuroscientific implications. Neuron *107*, 603–616.

Britten, K.H., Shadlen, M.N., Newsome, W.T., and Movshon, J.A. (1992). The analysis of visual motion: a comparison of neuronal and psychophysical performance. J. Neurosci. *12*, 4745–4765.

Cadieu, C.F., Hong, H., Yamins, D.L., Pinto, N., Ardila, D., Solomon, E.A., Majaj, N.J., and DiCarlo, J.J. (2014). Deep neural networks rival the representation of primate IT cortex for core visual object recognition. PLoS Comput. Biol. *10*, e1003963.

Carandini, M., and Heeger, D.J. (2011). Normalization as a canonical neural computation. Nat. Rev. Neurosci. *13*, 51–62.

Carrasco, M. (2011). Visual attention: the past 25 years. Vision Res. *51*, 1484–1525.

Chaisangmongkon, W., Swaminathan, S.K., Freedman, D.J., and Wang, X.-J. (2017). Computing by robust transience: how the fronto-parietal network performs sequential, category-based decisions. Neuron *93*, 1504–1517.e4.

Chen, T.Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D.K. (2018). Neural ordinary differential equations. Adv. Neural Inf. Process. Syst. *31*, 6571–6583.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv, 1406.1078 https://arxiv.org/abs/1406.1078.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv, 1412.3555 https://arxiv.org/abs/1412.3555.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). arXiv, 1511.07289 https://arxiv.org/abs/1511.07289.

Cohen, J.D., Dunbar, K., and McClelland, J.L. (1990). On the control of automatic processes: a parallel distributed processing account of the Stroop effect. Psychol. Rev. *97*, 332–361.

Costa, R., Assael, I.A., Shillingford, B., de Freitas, N., and Vogels, T. (2017). Cortical microcircuits as gated-recurrent neural networks. Adv. Neural Inf. Process. Syst. *30*, 272–283.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv, 1602.02830 https://arxiv.org/abs/1602.02830.

Cueva, C.J., and Wei, X.-X. (2018). Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. arXiv, 1803.07770 https://arxiv.org/abs/1803.07770.

Daw, N.D., Gershman, S.J., Seymour, B., Dayan, P., and Dolan, R.J. (2011). Model-based influences on humans' choices and striatal prediction errors. Neuron *69*, 1204–1215.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition (IEEE), pp. 248–255.

Desimone, R., and Duncan, J. (1995). Neural mechanisms of selective visual attention. Annu. Rev. Neurosci. *18*, 193–222.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv, 1810.04805 https://arxiv.org/abs/1810.04805.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. *12*, 2121–2159.

Eliasmith, C., Stewart, T.C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., and Rasmussen, D. (2012). A large-scale model of the functioning brain. Science *338*, 1202–1205.

Elman, J.L. (1990). Finding structure in time. Cogn. Sci. *14*, 179–211.

Erhan, D., Bengio, Y., Courville, A., and Vincent, P. (2009). Visualizing higher-layer features of a deep network. University of Montreal *1341*, 1.

Felleman, D.J., and Van Essen, D.C. (1991). Distributed hierarchical processing in the primate cerebral cortex. Cereb. Cortex *1*, 1–47.

Freedman, D.J., and Assad, J.A. (2006). Experience-dependent representation of visual categories in parietal cortex. Nature *443*, 85–88.

Freeman, J., and Simoncelli, E.P. (2011). Metamers of the ventral stream. Nat. Neurosci. *14*, 1195–1201.

Fukushima, K., and Miyake, S. (1982). Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. Pattern Recognit. *15*, 455–469.

Fukushima, K., Miyake, S., and Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics (IEEE), pp. 826–834.

Fusi, S., Drew, P.J., and Abbott, L.F. (2005). Cascade models of synaptically stored memories. Neuron *45*, 599–611.

Gers, F.A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: continual prediction with LSTM. Neural Comput. *12*, 2451–2471.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics *15*, 315–323.

Gold, J.I., and Shadlen, M.N. (2007). The neural basis of decision making. Annu. Rev. Neurosci. *30*, 535–574.

Goldman-Rakic, P.S. (1995). Cellular basis of working memory. Neuron *14*, 477–485.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. Adv. Neural Inf. Process. Syst. *27*, 2672–2680.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning (MIT Press).

Goudar, V., and Buonomano, D.V. (2018). Encoding sensory and motor patterns as time-invariant trajectories in recurrent neural networks. eLife *7*, e31134.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. arXiv, 1410.5401 https://arxiv.org/abs/1410.5401.

Grutzendler, J., Kasthuri, N., and Gan, W.-B. (2002). Long-term dendritic spine stability in the adult cortex. Nature *420*, 812–816.

Guerguiev, J., Lillicrap, T.P., and Richards, B.A. (2017). Towards deep learning with segregated dendrites. eLife *6*, e22901.

Haroush, K., and Williams, Z.M. (2015). Neuronal prediction of opponent's behavior during cooperative social interchange in primates. Cell *160*, 1233–1245.

Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. Neuron *95*, 245–258.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In 2015 IEEE International Conference on Computer Vision (IEEE), pp. 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (IEEE), pp. 770–778.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In 2017 IEEE International Conference on Computer Vision (IEEE), pp. 2961–2969.

Hebb, D.O. (2005). The Organization of Behavior: A Neuropsychological Theory (Psychology Press).

Heilbron, M., and Chait, M. (2018). Great expectations: is there evidence for predictive coding in auditory cortex? Neuroscience *389*, 54–73.

Helmstaedter, M., Briggman, K.L., Turaga, S.C., Jain, V., Seung, H.S., and Denk, W. (2013). Connectomic reconstruction of the inner plexiform layer in the mouse retina. Nature *500*, 168–174.

Hénaff, O.J., Goris, R.L.T., and Simoncelli, E.P. (2019). Perceptual straightening of natural videos. Nat. Neurosci. *22*, 984–991.

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. Neural Comput. *9*, 1735–1780.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. USA *79*, 2554–2558.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Netw. *2*, 359–366.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K.Q. (2017). Densely connected convolutional networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (IEEE), pp. 4700–4708.

Hubel, D.H., and Wiesel, T.N. (1959). Receptive fields of single neurones in the cat's striate cortex. J. Physiol. *148*, 574–591.

Hubel, D.H., and Wiesel, T.N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J. Physiol. *160*, 106–154.

Huh, D., and Sejnowski, T.J. (2018). Gradient descent for spiking neural networks. Adv. Neural Inf. Process. Syst. *31*, 1433–1443.

Ioffe, S., and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv, 1502.03167 https://arxiv.org/abs/1502.03167.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. Adv. Neural Inf. Process. Syst. *31*, 8571–8580.

Jaeger, H., and Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science *304*, 78–80.

Januszewski, M., Kornfeld, J., Li, P.H., Pope, A., Blakely, T., Lindsey, L., Maitin-Shepard, J., Tyka, M., Denk, W., and Jain, V. (2018). High-precision automated reconstruction of neurons with flood-filling networks. Nat. Methods *15*, 605–610.

Jones, J.P., and Palmer, L.A. (1987). An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. J. Neurophysiol. *58*, 1233–1258.

Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In ISCA '17: Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA), pp. 1–12.

Kaplanis, C., Shanahan, M., and Clopath, C. (2018). Continual reinforcement learning with complex synapses. arXiv, 1802.07239 https://arxiv.org/abs/1802.07239.

Kar, K., Kubilius, J., Schmidt, K., Issa, E.B., and DiCarlo, J.J. (2019). Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. Nat. Neurosci. *22*, 974–983.

Khaligh-Razavi, S.-M., and Kriegeskorte, N. (2014). Deep supervised, but not unsupervised, models may explain IT cortical representation. PLoS Comput. Biol. *10*, e1003915.

Kiani, R., and Shadlen, M.N. (2009). Representation of confidence associated with a decision by neurons in the parietal cortex. Science *324*, 759–764.

Kietzmann, T.C., Spoerer, C.J., Sörensen, L.K.A., Cichy, R.M., Hauk, O., and Kriegeskorte, N. (2019). Recurrence is required to capture the representational dynamics of the human visual system. Proc. Natl. Acad. Sci. USA *116*, 21854–21863.

Kingma, D.P., and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv, 1412.6980 https://arxiv.org/abs/1412.6980.

Kingma, D.P., and Welling, M. (2013). Auto-Encoding Variational Bayes. arXiv, 1312.6114 https://arxiv.org/abs/1312.6114.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. Proc. Natl. Acad. Sci. USA *114*, 3521–3526.

Kobak, D., Brendel, W., Constantinidis, C., Feierstein, C.E., Kepecs, A., Mainen, Z.F., Qi, X.-L., Romo, R., Uchida, N., and Machens, C.K. (2016). Demixed principal component analysis of neural population data. eLife *5*, e10989.

Koch, C., and Ullman, S. (1987). Shifts in selective visual attention: towards the underlying neural circuitry. In Matters of Intelligence, L.M. Vaina, ed. (Springer), pp. 115–141.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. (2019). Similarity of Neural Network Representations Revisited. arXiv, 1905.00414 https://arxiv.org/abs/1905.00414.

Kriegeskorte, N. (2015). Deep neural networks: a new framework for modeling biological vision and brain information processing. Annu. Rev. Vis. Sci. *1*, 417–446.

Kriegeskorte, N., Mur, M., and Bandettini, P. (2008). Representational similarity analysis - connecting the branches of systems neuroscience. Front. Syst. Neurosci. *2*, 4.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. *25*, 1097–1105.

Krogh, A., and Hertz, J.A. (1992). A simple weight decay can improve generalization. Adv. Neural Inf. Process. Syst. *4*, 950–957.

Kuffler, S.W. (1953). Discharge patterns and functional organization of mammalian retina. J. Neurophysiol. *16*, 37–68.

Laje, R., and Buonomano, D.V. (2013). Robust timing and motor patterns by taming chaos in recurrent neural networks. Nat. Neurosci. *16*, 925–933.

Le, Q.V., Jaitly, N., and Hinton, G.E. (2015). A simple way to initialize recurrent networks of rectified linear units. arXiv, 1504.00941 https://arxiv.org/abs/1504.00941.

LeCun, Y. (1988). A theoretical framework for back-propagation. In Proceedings of the 1988 Connectionist Models Summer School, D. Touretzky, G. Hinton, and T. Sejnowski, eds. (Morgan Kaufmann), pp. 21–28.

LeCun, Y., and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. In The Handbook of Brain Theory and Neural Networks, M.A. Arbib, ed. (MIT Press), pp. 255–258.

LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., and Jackel, L.D. (1990). Handwritten digit recognition with a back-propagation network. Adv. Neural Inf. Process. Syst. *2*, 396–404.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proc. IEEE *86*, 2278–2324.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature *521*, 436–444.

Lillicrap, T.P., Cownden, D., Tweed, D.B., and Akerman, C.J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. Nat. Commun. *7*, 13276.

Lillicrap, T.P., Santoro, A., Marris, L., Akerman, C.J., and Hinton, G. (2020). Backpropagation and the brain. Nat. Rev. Neurosci. *21*, 335–346.

Lindsay, G.W. (2020). Attention in psychology, neuroscience, and machine learning. Front. Comput. Neurosci. *14*, 29.

Lindsay, G.W., and Miller, K.D. (2018). How biological attention mechanisms improve task performance in a large-scale visual system model. eLife *7*, e38105.

Lindsey, J., Ocko, S.A., Ganguli, S., and Deny, S. (2019). A unified theory of early visual representations from retina to cortex through anatomically constrained deep cnns. arXiv, 1901.00945 https://arxiv.org/abs/1901.00945.

Lotter, W., Kreiman, G., and Cox, D. (2016). Deep predictive coding networks for video prediction and unsupervised learning. arXiv, 1605.08104 https://arxiv.org/abs/1605.08104.

Maheswaranathan, N., Williams, A.H., Golub, M.D., Ganguli, S., and Sussillo, D. (2019). Universality and individuality in neural dynamics across large populations of recurrent networks. arXiv, 1907.08549 https://arxiv.org/abs/1907.08549.

Mante, V., Sussillo, D., Shenoy, K.V., and Newsome, W.T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature *503*, 78–84.

Markov, N.T., Ercsey-Ravasz, M.M., Ribeiro Gomes, A.R., Lamy, C., Magrou, L., Vezoli, J., Misery, P., Falchier, A., Quilodran, R., Gariel, M.A., et al. (2014). A weighted and directed interareal connectivity matrix for macaque cerebral cortex. Cereb. Cortex *24*, 17–36.

Markram, H., Wang, Y., and Tsodyks, M. (1998). Differential signaling via the same axon of neocortical pyramidal neurons. Proc. Natl. Acad. Sci. USA *95*, 5323–5328.

Masse, N.Y., Grant, G.D., and Freedman, D.J. (2018). Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. Proc. Natl. Acad. Sci. USA *115*, E10467–E10475.

Masse, N.Y., Yang, G.R., Song, H.F., Wang, X.-J., and Freedman, D.J. (2019). Circuit mechanisms for the maintenance and manipulation of information in working memory. Nat. Neurosci. *22*, 1159–1167.

Mastrogiuseppe, F., and Ostojic, S. (2018). Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. Neuron *99*, 609–623.e29.

Mathis, A., Mamidanna, P., Cury, K.M., Abe, T., Murthy, V.N., Mathis, M.W., and Bethge, M. (2018). DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. Nat. Neurosci. *21*, 1281–1289.

McCloskey, M., and Cohen, N.J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation *24*, 109–165.

McIntosh, L.T., Maheswaranathan, N., Nayebi, A., Ganguli, S., and Baccus, S.A. (2016). Deep learning models of the retinal response to natural scenes. Adv. Neural Inf. Process. Syst. *29*, 1369–1377.

Merolla, P.A., Arthur, J.V., Alvarez-Icaza, R., Cassidy, A.S., Sawada, J., Akopyan, F., Jackson, B.L., Imam, N., Guo, C., Nakamura, Y., et al. (2014). Artificial brains. A million spiking-neuron integrated circuit with a scalable communication network and interface. Science *345*, 668–673.

Metz, L., Maheswaranathan, N., Cheung, B., and Sohl-Dickstein, J. (2018). Meta-learning update rules for unsupervised representation learning. arXiv, 1804.00222 https://arxiv.org/abs/1804.00222.

Miconi, T., Clune, J., and Stanley, K.O. (2018). Differentiable plasticity: training plastic neural networks with backpropagation. arXiv, 1804.02464 https://arxiv.org/abs/1804.02464.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. Nature *518*, 529–533.

Mongillo, G., Barak, O., and Tsodyks, M. (2008). Synaptic theory of working memory. Science *319*, 1543–1546.

Murray, J.M. (2019). Local online learning in recurrent networks with random feedback. eLife *8*, e43299.

Nath, T., Mathis, A., Chen, A.C., Patel, A., Bethge, M., and Mathis, M.W. (2019). Using DeepLabCut for 3D markerless pose estimation across species and behaviors. Nat. Protoc. *14*, 2152–2176.

Nayebi, A., Bear, D., Kubilius, J., Kar, K., Ganguli, S., Sussillo, D., DiCarlo, J.J., and Yamins, D.L. (2018). Task-driven convolutional recurrent models of the visual system. Adv. Neural Inf. Process. Syst. *31*, 5290–5301.

Nicola, W., and Clopath, C. (2017). Supervised learning in spiking neural networks with FORCE training. Nat. Commun. *8*, 2208.

Niv, Y. (2009). Reinforcement learning in the brain. J. Math. Psychol. *53*, 139–154.

Oh, S.W., Harris, J.A., Ng, L., Winslow, B., Cain, N., Mihalas, S., Wang, Q., Lau, C., Kuan, L., Henry, A.M., et al. (2014). A mesoscale connectome of the mouse brain. Nature *508*, 207–214.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. J. Math. Biol. *15*, 267–273.

Olsen, S.R., Bortone, D.S., Adesnik, H., and Scanziani, M. (2012). Gain control by layer six in cortical circuits of vision. Nature *483*, 47–52.

Olshausen, B.A., and Field, D.J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature *381*, 607–609.

Orhan, A.E., and Ma, W.J. (2019). A diverse range of factors affect the nature of neural representations underlying short-term memory. Nat. Neurosci. *22*, 275–283.

Pandarinath, C., O'Shea, D.J., Collins, J., Jozefowicz, R., Stavisky, S.D., Kao, J.C., Trautmann, E.M., Kaufman, M.T., Ryu, S.I., Hochberg, L.R., et al. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. Nat. Methods *15*, 805–815.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. Proceedings of the 30th International Conference on Machine Learning 28, pp. 1310–1318.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. *32*, 8024–8035.

Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., Wang, G., Zou, Z., Wu, Z., He, W., et al. (2019). Towards artificial general intelligence with hybrid Tianjic chip architecture. Nature *572*, 106–111.

Polyak, B.T. (1964). Some methods of speeding up the convergence of iteration methods. USSR Comput. Math. Math. Phys. *4*, 1–17.

Ponce, C.R., Xiao, W., Schade, P.F., Hartmann, T.S., Kreiman, G., and Livingstone, M.S. (2019). Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences. Cell *177*, 999–1009.e10.

Prenger, R., Wu, M.C.-K., David, S.V., and Gallant, J.L. (2004). Nonlinear V1 responses to natural scenes revealed by neural network analysis. Neural Netw. *17*, 663–679.

Rajalingham, R., Issa, E.B., Bashivan, P., Kar, K., Schmidt, K., and DiCarlo, J.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. *38*, 7255–7269.

Rajan, K., Harvey, C.D., and Tank, D.W. (2016). Recurrent network models of sequence generation and memory. Neuron *90*, 128–142.

Rao, R.P., and Ballard, D.H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. Nat. Neurosci. *2*, 79–87.

Reynolds, J.H., and Heeger, D.J. (2009). The normalization model of attention. Neuron *61*, 168–185.

Richards, B.A., Lillicrap, T.P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R.P., de Berker, A., Ganguli, S., et al. (2019). A deep learning framework for neuroscience. Nat. Neurosci. *22*, 1761–1770.

Riesenhuber, M., and Poggio, T. (1999). Hierarchical models of object recognition in cortex. Nat. Neurosci. *2*, 1019–1025.

Rigotti, M., Ben Dayan Rubin, D., Wang, X.-J., and Fusi, S. (2010). Internal representation of task rules by recurrent dynamics: the importance of the diversity of neural responses. Front. Comput. Neurosci. *4*, 24.

Rigotti, M., Barak, O., Warden, M.R., Wang, X.-J., Daw, N.D., Miller, E.K., and Fusi, S. (2013). The importance of mixed selectivity in complex cognitive tasks. Nature *497*, 585–590.

Robbins, H., and Monro, S. (1951). A stochastic approximation method. Ann. Math. Stat. *22*, 400–407.

Roelfsema, P.R., and Holtmaat, A. (2018). Control of synaptic plasticity in deep cortical networks. Nat. Rev. Neurosci. *19*, 166–180.

Roitman, J.D., and Shadlen, M.N. (2002). Response of neurons in the lateral intraparietal area during a combined visual discrimination reaction time task. J. Neurosci. *22*, 9475–9489.

Romo, R., Brody, C.D., Hernández, A., and Lemus, L. (1999). Neuronal correlates of parametric working memory in the prefrontal cortex. Nature *399*, 470–473.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. *65*, 386–408.

Rosenblatt, F. (1962). Principles of neurodynamics: Perceptions and the theory of brain mechanisms. In Brain Theory, G. Palm and A. Aertsen, eds. (Springer), pp. 245–248.

Rubin, D.B., Van Hooser, S.D., and Miller, K.D. (2015). The stabilized supralinear network: a unifying circuit motif underlying multi-input integration in sensory cortex. Neuron *85*, 402–417.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning representations by back-propagating errors. Nature *323*, 533–536.

Sacramento, J., Costa, R.P., Bengio, Y., and Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. Adv. Neural Inf. Process. Syst. *31*, 8721–8732.

Salinas, E., and Thier, P. (2000). Gain modulation: a major computational principle of the central nervous system. Neuron *27*, 15–21.

Saxe, A.M., McClelland, J.L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv, 1312.6120 https://arxiv.org/abs/1312.6120.

Saxe, A.M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B.D., and Cox, D.D. (2019a). On the information bottleneck theory of deep learning. J. Stat. Mech. *2019*, 124020.

Saxe, A.M., McClelland, J.L., and Ganguli, S. (2019b). A mathematical theory of semantic development in deep neural networks. Proc. Natl. Acad. Sci. USA *116*, 11537–11546.

Schultz, W., Dayan, P., and Montague, P.R. (1997). A neural substrate of prediction and reward. Science *275*, 1593–1599.

Seung, H.S. (1996). How the brain keeps the eyes still. Proc. Natl. Acad. Sci. USA *93*, 13339–13344.

Shu, Y., Hasenstaub, A., and McCormick, D.A. (2003). Turning on and off recurrent balanced cortical activity. Nature *423*, 288–293.

Shwartz-Ziv, R., and Tishby, N. (2017). Opening the black box of deep neural networks via information. arXiv, 1703.00810 https://arxiv.org/abs/1703.00810.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of Go without human knowledge. Nature *550*, 354–359.

Simonyan, K., and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv, 1409.1556 https://arxiv.org/abs/1409.1556.

Sompolinsky, H., Crisanti, A., and Sommers, H.-J. (1988). Chaos in random neural networks. Phys. Rev. Lett. *61*, 259–262.

Song, S., Miller, K.D., and Abbott, L.F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. Nat. Neurosci. *3*, 919–926.

Song, H.F., Yang, G.R., and Wang, X.-J. (2016). Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. PLoS Comput. Biol. *12*, e1004792.

Song, H.F., Yang, G.R., and Wang, X.-J. (2017). Reward-based training of recurrent neural networks for cognitive and value-based tasks. eLife *6*, e21492.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. *15*, 1929–1958.

Stokes, M.G. (2015). 'Activity-silent' working memory in prefrontal cortex: a dynamic coding framework. Trends Cogn. Sci. *19*, 394–405.

Strogatz, S.H. (2001). Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering (Studies in Nonlinearity) (CRC Press).

Sussillo, D. (2014). Neural circuits as computational dynamical systems. Curr. Opin. Neurobiol. *25*, 156–163.

Sussillo, D., and Abbott, L.F. (2009). Generating coherent patterns of activity from chaotic neural networks. Neuron *63*, 544–557.

Sussillo, D., and Barak, O. (2013). Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. Neural Comput. *25*, 626–649.

Sussillo, D., Churchland, M.M., Kaufman, M.T., and Shenoy, K.V. (2015). A neural network that finds a naturalistic solution for the production of muscle activity. Nat. Neurosci. *18*, 1025–1033.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. Proceedings of the 30th International Conference on Machine Learning *28*, 1139–1147.

Sutton, R.S., and Barto, A.G. (2018). Reinforcement Learning: An Introduction (MIT Press).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. arXiv, 1312.6199 https://arxiv.org/abs/1312.6199.

Tavanaei, A., Ghodrati, M., Kheradpisheh, S.R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. Neural Netw. *111*, 47–63.

Tieleman, T., and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning 4, pp. 26–31.

Tikhonov, A.N. (1943). On the stability of inverse problems. Dokl. Akad. Nauk SSSR *39*, 195–198.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. arXiv, 1607.08022 https://arxiv.org/abs/1607.08022.

van Vreeswijk, C., and Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. Science *274*, 1724–1726.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. Adv. Neural Inf. Process. Syst. *30*, 5998–6008.

Wang, X.-J. (2001). Synaptic reverberation underlying mnemonic persistent activity. Trends Neurosci. *24*, 455–463.

Wang, X.-J. (2002). Probabilistic decision making by slow reverberation in cortical circuits. Neuron *36*, 955–968.

Wang, X.-J. (2008). Decision making in recurrent neuronal circuits. Neuron *60*, 215–234.

Wang, X.-J., and Yang, G.R. (2018). A disinhibitory circuit motif and flexible information routing in the brain. Curr. Opin. Neurobiol. *49*, 75–83.

Wang, X.-J., Tegnér, J., Constantinidis, C., and Goldman-Rakic, P.S. (2004). Division of labor among distinct subtypes of inhibitory neurons in a cortical microcircuit of working memory. Proc. Natl. Acad. Sci. USA *101*, 1368–1373.

Wang, J., Narain, D., Hosseini, E.A., and Jazayeri, M. (2018). Flexible timing by temporal scaling of cortical responses. Nat. Neurosci. *21*, 102–110.

Werbos, P.J. (1990). Backpropagation through time: what it does and how to do it. Proc. IEEE *78*, 1550–1560.

Williams, R.J., and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. Neural Comput. *1*, 270–280.

Williams, A.H., Kim, T.H., Wang, F., Vyas, S., Ryu, S.I., Shenoy, K.V., Schnitzer, M., Kolda, T.G., and Ganguli, S. (2018). Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. Neuron *98*, 1099–1115.e8.

Wilson, H.R., and Cowan, J.D. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. Biophys. J. *12*, 1–24.

Wu, Y., and He, K. (2018). Group normalization. In Computer Vision – ECCV 2018, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds. (ECCV), pp. 3–19.

Xie, X., and Seung, H.S. (2003). Equivalence of backpropagation and contrastive Hebbian learning in a layered network. Neural Comput. *15*, 441–454.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. Proceedings of the 32nd International Conference on Machine Learning 37, pp. 2048–2057.

Yamane, Y., Carlson, E.T., Bowman, K.C., Wang, Z., and Connor, C.E. (2008). A neural code for three-dimensional object shape in macaque inferotemporal cortex. Nat. Neurosci. *11*, 1352–1360.

Yamins, D.L., and DiCarlo, J.J. (2016). Using goal-driven deep learning models to understand sensory cortex. Nat. Neurosci. *19*, 356–365.

Yamins, D.L., Hong, H., Cadieu, C.F., Solomon, E.A., Seibert, D., and DiCarlo, J.J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. Proc. Natl. Acad. Sci. USA *111*, 8619–8624.

Yang, G., Pan, F., and Gan, W.-B. (2009). Stably maintained dendritic spines are associated with lifelong memories. Nature *462*, 920–924.

Yang, G.R., Murray, J.D., and Wang, X.-J. (2016). A dendritic disinhibitory circuit mechanism for pathway-specific gating. Nat. Commun. *7*, 12815.

Yang, G.R., Ganichev, I., Wang, X.-J., Shlens, J., and Sussillo, D. (2018). A dataset and architecture for visual reasoning with a working memory. In Computer Vision – ECCV 2018, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds. (ECCV), pp. 729–745.

Yang, G.R., Joglekar, M.R., Song, H.F., Newsome, W.T., and Wang, X.-J. (2019). Task representations in neural networks trained to perform many cognitive tasks. Nat. Neurosci. *22*, 297–306.

Zeiler, M.D., and Fergus, R. (2014). Visualizing and understanding convolutional networks. In Computer Vision – ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds. (ECCV), pp. 818–833.

Zenke, F., and Ganguli, S. (2018). Superspike: Supervised learning in multilayer spiking neural networks. Neural Comput. *30*, 1514–1541.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. Proceedings of the 34th International Conference on Machine Learning 70, pp. 3987–3995.

Zhuang, C., Yan, S., Nayebi, A., and Yamins, D. (2019). Self-supervised neural network models of higher visual cortex development. In 2019 Conference on Cognitive Computational Neuroscience (CCN), pp. 566–569.

Zipser, D., and Andersen, R.A. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. Nature *331*, 679–684.

Zucker, R.S., and Regehr, W.G. (2002). Short-term synaptic plasticity. Annu. Rev. Physiol. *64*, 355–405.

Update

Neuron

Volume 109, Issue 4, 17 February 2021, Page 739

DOI: https://doi.org/10.1016/j.neuron.2021.01.022

Correction

Artificial neural networks for neuroscientists: a primer

Guangyu Robert Yang* and Xiao-Jing Wang*
*Correspondence: robert.yang@columbia.edu (G.R.Y.), xjwang@nyu.edu (X.-J.W.)

https://doi.org/10.1016/j.neuron.2021.01.022

(Neuron *107*, 1048–1070; September 23, 2020)

We found out that we have used an outdated version of Figure 8 during the final figure submission process. In the corrected Figure 8D, the input gate (red) of the neural network should stay low at value 0 and only become activated at value 1 when the *memorize input* signal (Figure 8B, red) is activated, matching the description in the figure legend. The correct Figure 8D has been used in our arXiv paper and has now been updated in the published paper online. Its source code has been available in our GitHub repository. The authors apologize for any confusion the errors may have caused.

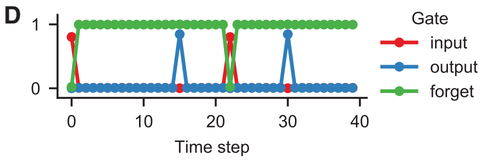

Figure 8D. Visualizing LSTM activity in a simple memory task (corrected)

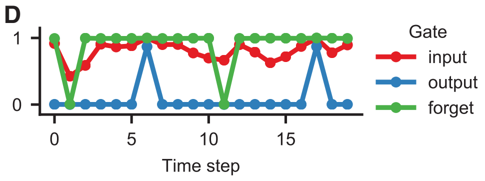

Figure 8D. Visualizing LSTM activity in a simple memory task (original)