Locations in the Neocortex: A Theory of Sensorimotor Object Recognition Using Cortical Grid Cells

Marcus Lewis, Scott Purdy, Subutai Ahmad, and Jeff Hawkins
Numenta, Inc., Redwood City, CA, USA

ABSTRACT

The neocortex is capable of modeling complex objects through sensorimotor interaction but the neural mechanisms are poorly understood. Grid cells in the entorhinal cortex represent the location of an animal in its environment, and this location is updated through movement and path integration. In this paper, we propose that grid-like cells in the neocortex represent the location of sensors on an object. We describe a two-layer neural network model that uses cortical grid cells and path integration to robustly learn and recognize objects through movement. Grid cells exhibit regular tiling over environments and are organized into modules, each with its own scale and orientation. A single module encodes position within the spatial scale of the module but is ambiguous over larger spaces. A set of modules can uniquely encode many large spaces. In our model, a layer of cells consisting of several grid-like modules represents a location in the reference frame of a specific object. Another layer of cells which processes sensory input receives this location input as context and uses it to encode the sensory input in the object's reference frame. Sensory input causes the network to invoke previously learned locations that are consistent with the input, and motor input causes the network to update those locations. Simulations show that the model can learn hundreds of objects even when object features alone are insufficient for disambiguation. We discuss the relationship of the model to cortical circuitry and suggest that the reciprocal connections between layers 4 and 6 fit the requirements of the model. We propose that the subgranular layers of cortical columns employ grid cell like mechanisms to represent object specific locations that are updated through movement.

INTRODUCTION

Our brains learn about the outside world by processing our sensory inputs and movements. As we touch an object, survey a visual scene, or explore an environment, the brain receives a series of sensations and movements, a *sensorimotor sequence*.

It's not clear how our brains extract reusable information from sensorimotor sequences. If the brain ignores the motor stream and learns objects using only the sensory stream, it will lose the ability to correctly combine information from multiple sensations. But if it learns objects by memorizing sensorimotor sequences, it won't be able to recognize familiar objects from novel sensorimotor sequences.

Most existing models of object recognition assume a strictly feedforward passive mechanism for object recognition (DiCarlo et al., 2012; Riesenhuber and Poggio, 1999) and it is unclear how sensorimotor information would be incorporated. Recent work from our lab (Hawkins et al., 2017) proposed that the neocortex processes a sensorimotor sequence by converting it into a sequence of *sensory features at object-centric locations*. The neocortex could then learn and recognize objects as sets of these sensory features at locations. This approach integrates movement into object recognition, and forms representations of objects that generalize over novel sequences of movement. However in that paper we left open the neural mechanisms for computing such a location signal.

In this paper, we show how the neocortex could compute these object-centric locations. With this missing piece filled in, we present a neural network model that learns to recognize static objects, receiving only a sensorimotor sequence as input.

Representing the locations of sensed features in object centric coordinates makes object recognition very efficient, but it requires a complex computation (Marr and Nishihara, 1978). The system must establish a coordinate system before it can begin representing the locations of sensed features. We show how part of this problem is simplified by using a conception of *location* that is inspired by grid cells. Recognizing an object requires establishing the directions of the axes of the coordinate system, but it doesn't require choosing an origin for the coordinate system. Our proposed model recognizes objects using the relative locations of features instead of using their locations relative to some origin.

Grid cell location representations fit naturally into neural mechanisms for processing sensorimotor sequences. Our model assigns each new object its own unique set of locations by first activating a random location representation and then updating it with each motor input. It recognizes objects by using each sensory input to recall and refine possible locations, and by using each motor input to update the possible locations.

We review the basic properties of grid cells and we propose that the neocortex uses analogs of grid cells to model objects just as the hippocampal formation uses them to model environments. Building on the theoretical framework introduced in (Hawkins et al., 2017) we propose that every neocortical column contains a variant of this model. Based on the cortical anatomy, we propose that cells in Layer 6 employ grid cell like mechanisms to represent object specific locations that are updated through movement. We propose that Layer 4 uses its input from Layer 6 to predict sensory input.

How Grid Cells Represent Locations and Movement

We first review how grid cells in the entorhinal cortex represent space and location. Although many details of grid cell function remain unknown, general consensus has emerged for a number of principles. Here we focus on two properties that are critical to our model: location coding and path integration.

Individual grid cells become active at multiple locations in an environment, typically in a repeating triangular lattice that

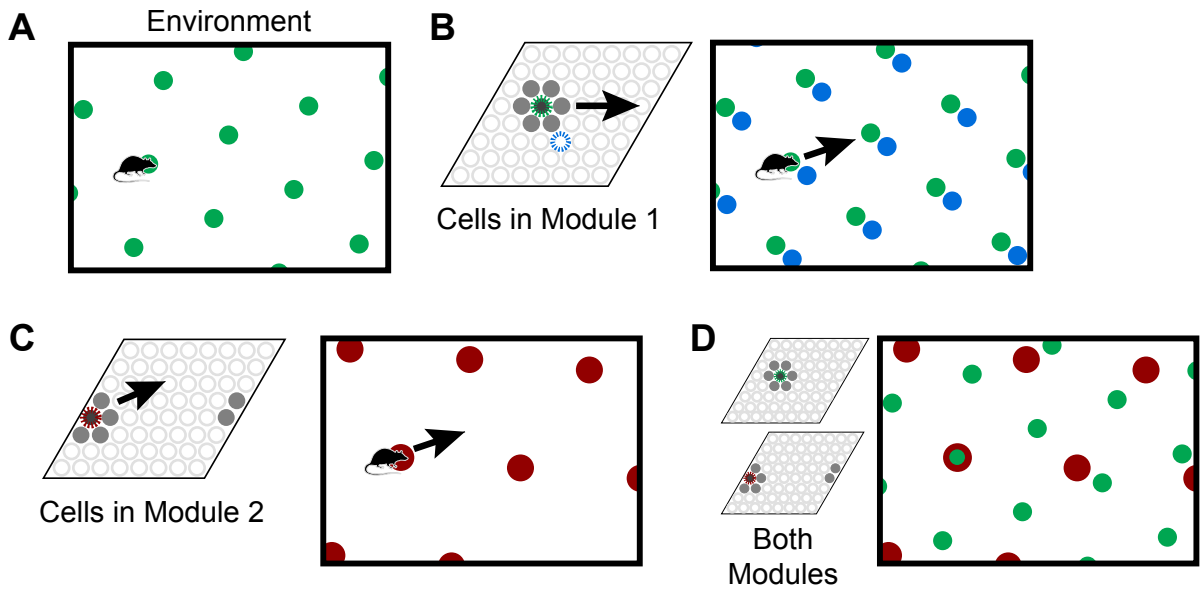

Figure 1. Grid cells represent locations in environments. **(A)** An individual grid cell becomes active at multiple locations (green circles) in an environment. The locations of activation form a repeating grid-like lattice. **(B)** A grid cell module (left) is a set of cells that share the same lattice scale and orientation but which activate at different relative positions in the environment. If you sort the cells by their relative firing locations, it forms a rhombus-shaped tile. As the animal moves, as shown by the arrow, a bump of cell activity will move in some direction through this rhombus. Two grid cells and their firing locations (green and blue) are highlighted. The grid cell module will activate cells at every location in an environment, but because of tiling, a single grid cell module cannot represent a location uniquely. **(C)** This figure shows how a second module tiles the same space differently. Each cell's firing fields have a larger scale and a different orientation than the module in (A) and (B). The same movement of the animal as shown by the arrow causes the bump to move in a different direction and a different distance than the bump in the first module in (B). In this case, the bump overlaps the edge of the rhombus, so it wraps around. **(D)** Although a single module cannot represent locations in an environment uniquely, the activity across multiple modules can. Here we superimpose the firing patterns of the two modules. Note that when the green and red cells fire together, only one location is possible. The larger the number of modules, the more locations that can be represented uniquely.

resembles a grid (**Figure 1A**). The side length of these triangles in is known as the grid cell's "scale". A grid cell "module" is a set of grid cells that activate with the same lattice scale and orientation but different positions, such that one or more grid cells will be active at any location (**Figure 1B**). If you sort the grid cells in a module by their relative firing locations, they form a rhombus-shaped tile. As the animal moves, a "bump" of activity moves across this rhombus (**Figure 1B and 1C**). The activity in a single module provides information on an animal's location, but this information is ambiguous; many locations within the environment can lead to the same activity.

To form a unique representation requires multiple grid cell modules with different scales or orientations (**Figure 1C and 1D**). For illustration purposes say we have 10 grid cell modules and each module can encode 25 possible locations via a bump of activity. These 10 bumps encode the current location of the animal. Notice, if the animal moves continuously in one direction the activity of individual modules will repeat due to the tiling, but the ensemble activity of ten modules is unlikely to repeat due to the differences in scale and orientation between the modules. The representational capacity formed by such a code is large. In our example the number of unique locations that can be represented is $25^{10} \approx 10^{14}$. A review of the capacity and noise robustness of grid codes can be found in (Fiete et al., 2008; Sreenivasan and Fiete, 2011).

As an animal moves, the active grid cells in a module change to reflect the animal's updated location. This change occurs even if the animal is in the dark (Hafting et al., 2005), telling us that grid cells are updated using information about the animal's movement. This process, called "path integration", has the desirable property that regardless of the path of movement, when the animal returns to the same physical location, then the same grid cells will be active (**Figure 2A**). Path integration is imprecise so in learned environments sensory landmarks are used to "anchor" the grid cells and prevent the accumulation of path integration errors (McNaughton et al., 2006; Ocko et al., 2018).

A final important property is that the location representations can be unique to each environment. Suppose that upon first entering a new environment each grid cell module activates a random bump to represent the current location. Then all the location representations that the animal can move to in that environment will, with high probability, be unique to that environment. The initial random starting point thus implicitly defines a unique location space for each environment, including locations that have not yet been explicitly visited. Since each module independently integrates motion information, path integration properties automatically hold for each new environment. Consequently, path integration can be learned once for each module and then reused across all environments. The location space for each new environment will be a tiny subset of the full space of possible cell activities (in our

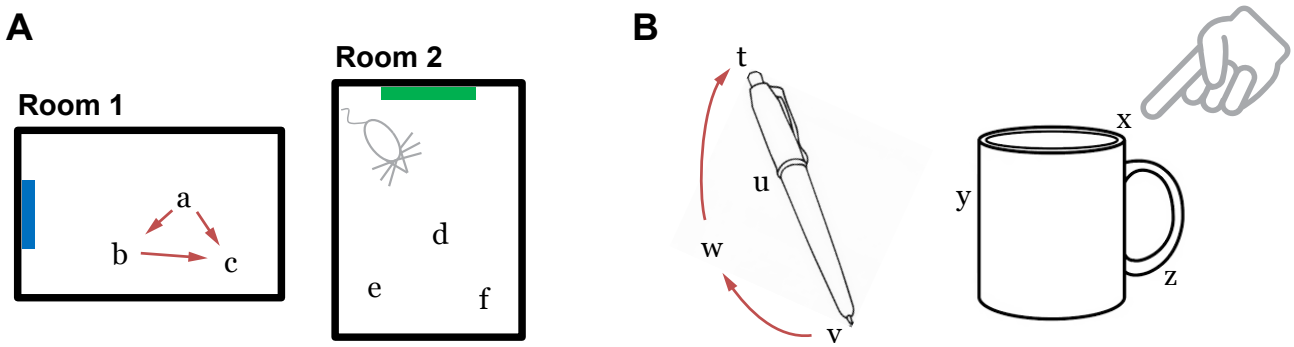

Figure 2. (A) Grid cells in the entorhinal cortex represent locations of a body in an environment. The location representations are updated by movement (Room 1). The path integration property ensures that the representation of location c is independent of the path taken to get there. Locations are unique to each environment such that the representations of locations a, b, and c are distinct from the representations of any point in Room 2. **(B)** We propose that the neocortex contains grid cell analogs that represent locations relative to an object. The location representations are unique to each object.

example above the full space contains 25^{10} points), thus the capacity for representing environments is quite large. When re-entering a previously learned environment, learned associations between sensory cues and grid cells are used to "re-anchor" or re-activate the previous location space.

To summarize the above properties, a set of grid cell modules can unambiguously represent locations in an environment. These locations can be path integrated via movement. By choosing random starting points within modules, unique location spaces can be defined for each environment. The space of all possible cell activations grows exponentially with the number of modules, thus the capacity for representing locations and environments is large.

MODEL

We propose that grid cell equivalents exist throughout the neocortex. Rather than representing the location of the animal in an environment, we propose that cortical grid cells represent the location of sensory patches, for example the tip of a finger, in the reference frame of an object (**Figure 2B**). Similar to traditional grid cells, cortical grid cells define a unique location space around each object. As a sensor moves, populations of grid cells representing each sensory patch's location will path integrate through unique location spaces. The relative locations of features on an object can thus be used as a powerful cue for disambiguating and recognizing objects.

Our network model integrates information over sensorimotor sequences, associating unique location spaces with objects and then identifying these location spaces. To outline the mechanism, let us first consider the question: how might a rat recognize a familiar environment? It must use its sensory input, but a single sensory observation is often insufficient to uniquely identify the environment. The rat thus needs to move and make multiple observations.

To combine information from multiple sensory observations, the rat could use each observation to recall the set of all locations associated with that feature. As it moves, it would then perform path integration to update each possible location. Subsequent sensory observations would be used to narrow down the set of locations and eventually disambiguate the location.

Our model uses this strategy to recognize objects with a moving sensor. According to this model, when you sense an object, the sensed feature causes you to recall locations where you've sensed this feature before. This is represented by a superposition of these previously learned locations. As you move your sensor, the network performs path integration on each of these recalled locations, i.e. the movement signal shifts the activity within each grid cell module. With subsequent sensations, the network will narrow down this list of locations until it uniquely identifies a specific location on a specific object that is consistent with the sequence of sensations and movements.

How does the animal use its sensory input to recall locations? It's unclear exactly how animals learn environments but sensory features are known to invoke grid cell activity associated with familiar environments (Barry et al., 2007). During learning our model associates sensory input with the currently active grid cells at each location.

How does the animal represent and perform path integration on an ambiguous location? In our model, the grid cell modules are capable of having multiple simultaneous bumps of cell activity. We refer to this set of simultaneously-active representations as a *union* of locations. The system is capable of path integrating unions of locations; during movement, every active bump in a module is shifted.

Model Description

Our two-layer model consists of two populations of neurons and four primary sets of connections **(Figure 3)**. For each movement of the sensor, the network goes through a progression of stages, processing the motor input followed by the sensory input. Each stage corresponds to using the connections from one of the numbered arrows in **Figure 3**. We show an example of the network going through these stages three times in **Figure 4**.

Stage 1. Motor input arrives before the sensory input and is processed by the location layer, which consists of grid cell modules. If this layer has an active location representation, it uses the motor

input to shift the activity in each module, computing the sensor's new location.

Stage 2. This updated grid cell activity propagates to the sensory layer and causes a set of predictions in that layer.

Stage 3. The sensory layer receives the actual sensory input. The predictions are combined with sensory input. The new activity is a union of highly sparse codes. Each sparse code represents a single sensory feature at a specific location that is consistent with the input so far.

Stage 4. The sensory layer activity propagates to the location layer. Each module activates a union of grid cells based on the sensory representation. The location layer will contain a union of representations of locations that are consistent with the input so far.

After the fourth stage the next motor action is initiated and the cycle repeats. The next few sections describe the network structure and each of these 4 stages in detail, as well as the learning process.

Notation and network structure

We compute the network activity through a set of discrete timesteps. Each time step t consists of a progression of the 4 stages. Each neuron in the network has a binary output; a cell is either active or inactive.

The location layer consists of a set of independent grid cell modules. The active cells of module i at time t are denoted by the binary array $\boldsymbol{A}_t^{\text{loc},i}$. The layer activity $\boldsymbol{A}_t^{\text{loc}}$ consists of the concatenation of all of the module activities $\boldsymbol{A}_t^{\text{loc},i}$. During inference, activity in the location layer is updated twice per timestep, once in response to movement and once in response to sensory-derived input. Where necessary, we denote these two vectors as $\boldsymbol{A}_{t,\text{move}}^{\text{loc}}$, and $\boldsymbol{A}_{t,\text{sense}}^{\text{loc}}$. Each module's bump of activity is randomly initialized for each new object, and the concatenated activity thus represents object-specific locations.

The sensory layer represents the sensed feature, and this representation is specific to an object-centric location. The layer is organized into a set of mini-columns such that all the cells in a mini-column share the same feedforward receptive fields and respond to the same sensory input (Hawkins and Ahmad, 2016). The active cells of mini-column i are denoted by the binary array $\boldsymbol{A}_t^{\text{in},i}$. The layer activity $\boldsymbol{A}_t^{\text{in}}$ consists of the concatenation of all of the mini-column activities $\boldsymbol{A}_t^{\text{in},i}$.

This model takes advantage of properties of active dendrites (Major et al., 2013). Every cell in each layer has a set of distal dendrites which are split into segments that each independently learn and activate in response to coincident patterns. $\boldsymbol{D}_{c,d}^{\text{loc}}$ and $\boldsymbol{D}_{c,d}^{\text{in}}$ each denote a vector which specifies the synapses of dendrite d on cell c in the location layer and sensory layer, respectively. The synapse weights are either 0 or 1.

The location layer projects to the distal dendrites of the sensory layer (**Figure 3**, connection 2). Thus $\boldsymbol{D}_{c,d}^{\text{in}}$ is a vector with the same length as $\boldsymbol{A}_t^{\text{loc}}$ where a 1 represents a connection to a cell in the location layer. These connections have a modulatory effect (see below). The sensory layer projects to the distal dendrites of the

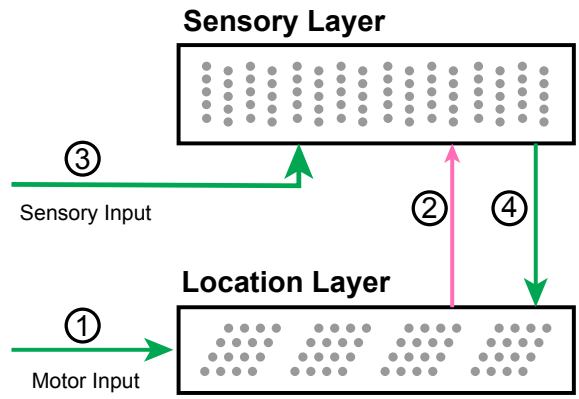

Figure 3. A diagram of the network with arrows indicating the main connections. (1) Motor input shifts the activity in the location layer. (2) The active location cells provide modulatory predictive input to the sensory layer. (3) Sensory input activates cells in the sensory layer. (4) The location is updated by the new sensory representation.

location layer (**Figure 3**, connection 4). Thus $\boldsymbol{D}_{c,d}^{\text{loc}}$ is a vector with the same length as $\boldsymbol{A}_t^{\text{in}}$ where a 1 represents a connection to a cell in the sensory layer. These vectors are generally extremely sparse as they connect to sparsely active cells during learning (see section on learning below).

In each timestep, dendritic segments that receive sufficient input exhibit a dendritic spike. Cells with dendritic spikes are denoted with the binary vectors $\boldsymbol{\pi}_t^{\text{in}}$ and $\boldsymbol{\pi}_t^{\text{loc}}$. These denote whether each cell was *predicted* from the other layer's activity. Designating θ^{in} and θ^{loc} as spike thresholds, and using the existential quantifier $\exists$,

$$\boldsymbol{\pi}_t^{\text{in},c} = \begin{cases} 1, & \exists_d[\boldsymbol{D}_{c,d}^{\text{in}} \cdot \boldsymbol{A}_{t,\text{move}}^{\text{loc}} \geq \theta^{\text{in}}] \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$\boldsymbol{\pi}_t^{\text{loc},c} = \begin{cases} 1, & \exists_d[\boldsymbol{D}_{c,d}^{\text{loc}} \cdot \boldsymbol{A}_t^{\text{in}} \geq \theta^{\text{loc}}] \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Stage 1: Use movement to update location layer

The location layer consists of a set of independent grid cell modules, each with its own scale and orientation. We follow notation and assumptions of other grid cell models and analyses (Ocko et al., 2018; Sreenivasan and Fiete, 2011). Within a module, the active cells are always part of a Gaussian bump of cell activity centered at a position, or *phase*, within the module's tile (**Figure 1B**). We designate the phase of module i's bump as $\vec{\phi}_t^i$. Each cell within a module is centered at a phase, and its activity at time t is proportional to its nearness to $\vec{\phi}_t^i$. The network responds to movement commands by updating $\vec{\phi}_t^i$ for each module.

Movement will shift each module's bump according to the module's scale and orientation. A 2D transform matrix $\boldsymbol{M}_i$ associated with each module represents how the module converts a movement vector for the sensor into a movement vector for the bump. Denoting the scale of module i as s_i and the orientation as θ_i, this transformation matrix is:

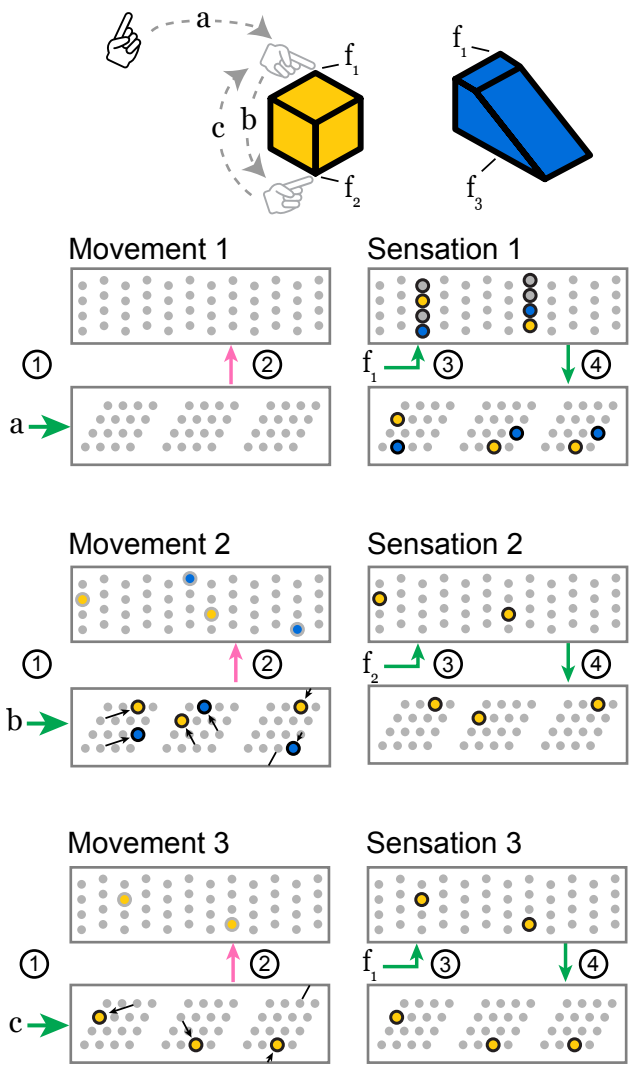

Figure 4. As the sensor moves over a previously learned object, these two layers receive a sensorimotor sequence and recognize the object. Features f_1, f_2 and f_3 indicate the sensory input invoked by touching the objects at the indicated locations. Motor commands a, b, and c indicate the motor input received by the network when the sensor makes a movement. The objects are colored to relate them to active cells below. We show three movements, each consisting of the four stages above, and we draw snapshots of the network at the end of stages 2 and 4. The stages 1 through 4 correspond to the connections in Figure 3. **Movement 1.** The network receives a movement command, and nothing happens because it doesn't have a current location representation. **Sensation 1.** The sensor senses feature f_1 which provides input to every cell in a set of mini-columns. None of the cells were predicted, so all become active. This feature has been learned on two objects, so this set of active cells contains two feature-at-location representations, shown in yellow and blue. These representations drive a pair of location representations to become active. This union of activity encodes the two possible locations that could have produced the sensation. **Movement 2.** Motor input b causes each module to perform path integration, shifting its bump according to its scale and orientation. The newly active cells provide modulatory input to the sensory layer which predicts two different potential features, f_2 and f_3, priming two representations to become active. **Sensation 2.** The sensor senses feature f_2, and only the predicted cells in the f_2 mini-columns become active. The other predicted cells do not activate. This active representation drives a single representation to become active in the location layer. At this point, the network has identified the cube. **Movement 3 and Sensation 3.** Subsequent movements maintain the unambiguous representation as long as the sensed features match those predicted by the path-integrated locations. A movement back to the original location, for instance, causes a prediction only for the f_1 representation specific to that location on the cube.

$$\boldsymbol{M}_i = \begin{bmatrix} s_i \cos\theta_i & s_i \cos(\theta_i + 60°) \\ s_i \sin\theta_i & s_i \sin(\theta_i + 60°) \end{bmatrix}^{-1} \tag{3}$$

Each module receives the same 2D movement vector $\vec{d}_t$, and it shifts its bump as follows:

$$\vec{\phi}^i_{t,\text{move}} = \left(\vec{\phi}^i_{t-1,\text{sense}} + \boldsymbol{M}_i \vec{d}_t\right) \bmod 1 \tag{4}$$

In addition to these properties, our model requires a grid cell module to be capable of path integrating multiple bumps simultaneously. Each module represents uncertainty by activating multiple bumps, one for each possible location. We refer to this as a *union* of locations. We designate a module's set of bumps as Φ^i_t. With every movement, we apply Eq. (4) to every phase in Φ^i_t.

$$\Phi^i_{t,\text{move}} = \left\{\left(\vec{\phi} + \boldsymbol{M}_i \vec{d}_t\right) \bmod 1 \;\middle|\; \vec{\phi} \in \Phi^i_{t-1,\text{sense}}\right\} \tag{5}$$

Rather than simulating individual cell dynamics explicitly, each module in the location layer simply maintains a list of activity bump phases and updates each one according to Eq. (5). The location layer then outputs the binary vector $\boldsymbol{A}^{\text{loc}}_{t,\text{move}}$ by thresholding the activity of each cell within the module (see Model Details). In the discussion we review models of individual grid cell dynamics and discuss their compatibility with unions.

Stage 2: Use updated location to form sensory predictions

In Stage 2, we compute which sensory features are predicted by the location layer. In our network these predictions are represented by $\boldsymbol{\pi}^{\text{in}}_t$, the activity of distal dendritic segments of cells in the sensory layer. We compute $\boldsymbol{\pi}^{\text{in}}_t$ from $\boldsymbol{A}^{\text{loc}}_{t,\text{move}}$ using Eq. (1) above.

Each sensory feature that has been encountered at any of the current possible locations will be predicted. Note that because $\boldsymbol{A}^{\text{loc}}_{t,\text{move}}$ is a concatenation of all grid cell modules, the predictions are based on highly specific location codes.

$\boldsymbol{\pi}^{\text{in}}_t$ has a modulatory effect on the activity of the sensory layer, as described in Stage 3.

Stage 3: Calculate activity in sensory layer
In Stage 3, the sensory layer uses the sensed feature to confirm correct predictions and to disregard incorrect predictions. When no predictions are correct, it activates all possible feature-at-location representations for the feature. This stage is responsible for both activating and narrowing unions.

The sensory layer is identical to the sensory input layer in (Hawkins et al., 2017). In this layer, all the cells in a mini-column share the same feedforward receptive fields. Each sensory feature is represented by a sparse subset of the mini-columns, denoted by $\boldsymbol{W}_t^{\text{in}}$. When a cell is predicted, it is primed to become active, and if it receives sensory input it will quickly activate and inhibit other cells in the mini-column.

The active cells within the sensory layer are selected by considering $\boldsymbol{W}_t^{\text{in}}$ and by considering which cells are predicted by the location layer, i.e. which cells have an active dendritic segment. If a cell is predicted and it is in a mini-column in $\boldsymbol{W}_t^{\text{in}}$, it becomes active. If a mini-column in $\boldsymbol{W}_t^{\text{in}}$ has no predicted cells, every cell in that mini-column becomes active.

$$\boldsymbol{A}_t^{in,i,c} = \begin{cases} 1, & i \in \boldsymbol{W}_t^{\text{in}} \text{ and } \boldsymbol{\pi}_t^{\text{in},i,c} > 0 \\ 1, & i \in \boldsymbol{W}_t^{\text{in}} \text{ and } \sum_{c'} \boldsymbol{\pi}_t^{\text{in},i,c'} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

If the location layer has uniquely identified the current location, there will be exactly one active cell in each mini-column in $\boldsymbol{W}_t^{\text{in}}$ encoding this sensory feature at the current location. If the location layer contains a union of locations, the sensory layer will represent this feature at a union of locations. Note that the location layer may predict features that are not represented in $\boldsymbol{W}_t^{\text{in}}$. These minicolumns will not contain any activity after this step (**Figure 4**, Sensation 2) and the set of possible objects is thus narrowed down.

Stage 4: Update location layer based on sensory cues
In Stage 4, the activity in the sensory layer recalls locations in the location layer. $\boldsymbol{A}_t^{\text{in}}$ is used to compute $\boldsymbol{\pi}_t^{\text{loc}}$ (Eq. (2)) which drives activity in the location layer. The list of activity bump phases in each module is replaced by a new set of bumps driven by sensory input. For each cell in the location layer that has a corresponding dendritic spike, the module activates a bump centered on that cell.

$$\Phi_{t,sense}^{loc,i} = \begin{cases} \{\vec{\phi}_c | c : \boldsymbol{\pi}_t^{\text{loc},i,c} > 0\}, \exists_c [\boldsymbol{\pi}_t^{\text{loc},i,c} > 0] \\ \Phi_{t,\text{move}}^{\text{loc},i}, & \text{otherwise} \end{cases} \tag{7}$$

This new set of bumps is often very similar to the previous set, as in **Figure 4,** Sensation 3. Note that this step happens during inference only. During learning, the location layer doesn't update in response to sensory input; we simply assign $\Phi_{t,\text{sense}}^{\text{loc},i} = \Phi_{t,\text{move}}^{\text{loc},i}$.

The active cells in the location layer $\boldsymbol{A}_{t,\text{sense}}^{\text{loc}}$ are computed from $\Phi_{t,\text{sense}}^{\text{loc}}$ (see Model Details).

Learning
In our model the learning process involves associating locations with sensory features, and vice versa. Learning occurs only on the distal dendritic segments. At the start of training on a new object, each module in the location layer activates a bump at a random phase. This instantiates a random location space. For the rest of training, we provide a sequence of motor and sensory inputs, calculating $\Phi_{t,\text{move}}^{\text{loc}}$ as before, shifting each module's bump with each movement.

Each sensory input is represented by a set of mini-columns $\boldsymbol{W}_{t,\text{sense}}^{\text{in}}$. Following Eq. (7) above, if this part of the object hasn't been learned yet there will be no predictions in the sensory layer and every cell in these mini-columns will become active. In this case a random cell in each active mini-column is selected as the cell to learn on, i.e. to represent this sensory input at this location. If this part of the object has been learned, there will be predictions in the sensory layer. In this case the cells corresponding to the existing active segments are selected to learn on. $\boldsymbol{A}_{t,\text{learn}}^{\text{in}}$ represents these learning cells for the current time step.

Each active cell in $\boldsymbol{A}_{t,\text{sense}}^{\text{loc}}$ and $\boldsymbol{A}_{t,\text{learn}}^{\text{in}}$ selects one of its dendritic segments d' and forms connections between this segment and each active cell in the other layer. Designating "|" as bitwise OR,

$$\boldsymbol{D}_{c,d'}^{\text{loc}} := \boldsymbol{D}_{c,d'}^{\text{loc}} \mid \boldsymbol{A}_{t,\text{learn}}^{\text{in}} \tag{8}$$

$$\boldsymbol{D}_{c,d'}^{\text{in}} := \boldsymbol{D}_{c,d'}^{\text{in}} \mid \boldsymbol{A}_{t,\text{sense}}^{\text{loc}} \tag{9}$$

SIMULATION RESULTS

We ran simulations to illustrate the ability of our model to recognize objects. We generated objects with varying numbers of shared features to test the ability of the network to disambiguate. Each object consisted of ten points chosen randomly from a four-by-four grid. We placed a feature at each point, choosing each feature randomly with replacement from a fixed feature pool. Features were shared across objects and a given feature could occur at multiple points on the same object.

For every simulation, we trained the network by visiting each point on each object once. For each point, we stored the activity in the location layer in a separate classifier. We then tested the network on each object by traversing each of the object points in random order. As the sensor traversed the object, we tested whether the location representation exactly matched the classifier's stored representation for that point on that object. If it matched this representation and if this representation was unique to this object, we considered the object to be recognized. If the network never converged to a single location representation after four complete passes over the object, or if it ever converged on a wrong location, we considered this a recognition failure. In most of our experiments, these were the two possible outcomes, but it's also possible for the network to converge on the correct location representation even if that location isn't unique to the object, for example if there aren't enough modules to create a unique code. In the sections ahead, we specifically note when this occurred.

We set the sensory layer to have 150 mini-columns and 16 cells per mini-column. Each sensory feature activated a predetermined, randomly selected set of 10 mini-columns. We varied the number of cells per module and the number of modules in the location layer. For these simulations we varied the module orientations but not the scales. The orientations were evenly spaced along the 60° range of

possible orientations. Each module used the same scale, and this scale was fixed to be less than the width of the objects. We set the dendritic thresholds θ^{loc} and θ^{in} to 8 and $\lceil n * 0.8 \rceil$, respectively, where n is the number of modules in the location layer and $\lceil\,\rceil$ is the ceiling operator.

Cell activity converges to a unique location representation.
We begin by demonstrating the cell activity in a typical recognition task, and we show how it varies as the network learns more objects. In **Figure 5** we show the actual grid cell activity for several modules in the location layer as the network sensed different objects. The network was first trained on 50 objects, each with ten features drawn from a pool of 40 features. The activity changes with each new sensation, first via path integration shifts based on the movement, followed by narrowing of the activity to only the locations consistent with the newly sensed feature. The network can take a different number of sensations to narrow to a single representation per module. Once the network has narrowed, the activity in a single module may be ambiguous but the set of active cells across all modules (only three of ten modules are shown) uniquely encode the object and location.

The recognition process always follows this template. The initial sensory input typically causes dense activation, assuming the sensory feature is not unique to a single location. With subsequent movement and sensation this activity becomes sparser and eventually converges on a single representation. In **Figure 6A** we aggregate the cell activity from **Figure 5** across all objects and all modules to show the average cell activation density after each sensation. As the network learns more objects, the initial density and the convergence time increase because the network recalls more locations-on-objects for each sensory input, and it has to disambiguate between more objects.

The model approximates an ideal observer.
Ideally, the network should recognize an object as soon as it receives a sensorimotor sequence that uniquely matches that object. In **Figure 6B** we compare the recognition time with such an ideal observer. The ideal detector stores both the features and their relative locations during training and exhaustively checks the current sensorimotor sequence against the stored objects during testing. It yields a correct classification as soon as the object is unambiguous based on the features and relative locations sensed up to that point. We also include a bag of features model. It ignores location information and yields a correct classification if it can unambiguously determine the object based solely on the sensory features.

This network uses 10 modules, and the dataset contains 100 objects with 10 unique features. Very few objects can possibly be determined by a single sensation, but three or four sensations are sufficient. As we increase the number of cells in each module, the model's performance approaches that of the ideal detector. With 40x40 cells per module, the two are near identical. The bag of features model often cannot uniquely identify the objects because many objects are different arrangements of identical sets of features.

Near the model's capacity limits, recognition time degrades.
As this model learns more objects, it eventually begins taking longer to recognize objects than the ideal observer. If it's pushed further, it eventually begins failing to recognize objects. In **Figure**

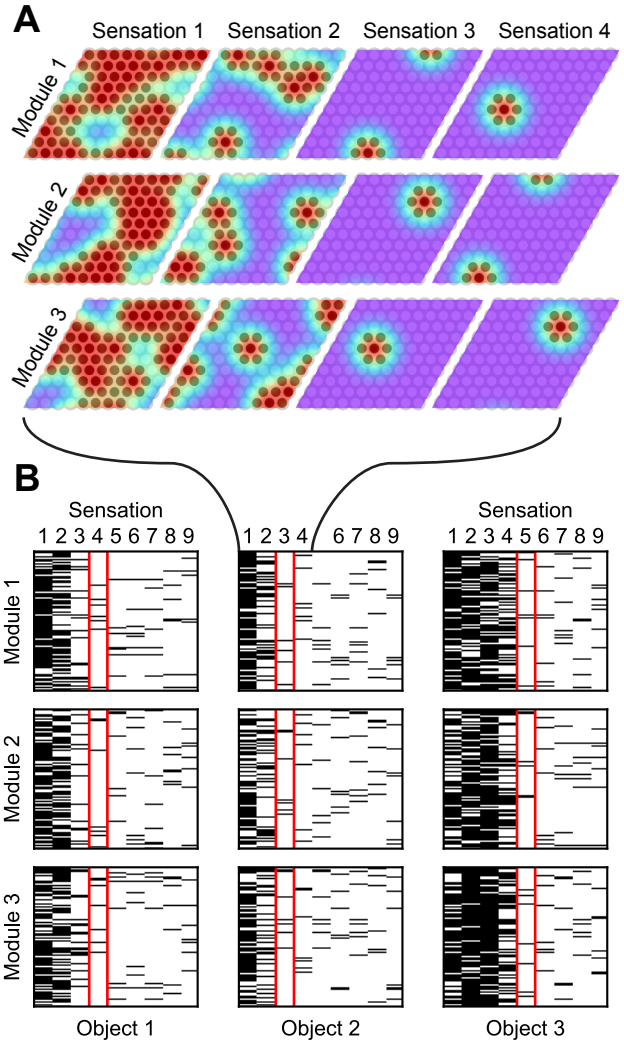

Figure 5. As the network recognizes an object, it converges onto a sparse activation. **(A)** Location layer cell activity in three (out of ten) grid cell modules while recognizing a single object. The bumps of activity are shown in color; red indicates a high firing rate. The location representation, shown as darkened circles, consists of cells that are sufficiently active. Each movement shifts the activity and each sensation narrows the activity to the cells that predict the new sensation. Cell activity converges onto a sparse representation by Sensation 3 and remains sparse afterward. **(B)** Cell activity in the same three modules as (A), shown for additional sensations and for two additional objects. Each module has 100 cells. The black lines indicate that the cell is active during the indicated sensation. After the first sensation, the location codes are very ambiguous in all cases. Depending on how common the sensed features are, the module activity narrows at different rates for the different objects. The sensation highlighted in red shows the first step in which the object is unambiguously determined. From this point on, the module activity shifts with each movement but remains unique to the object being sensed. (These simulations used a unique feature pool of size 40.)

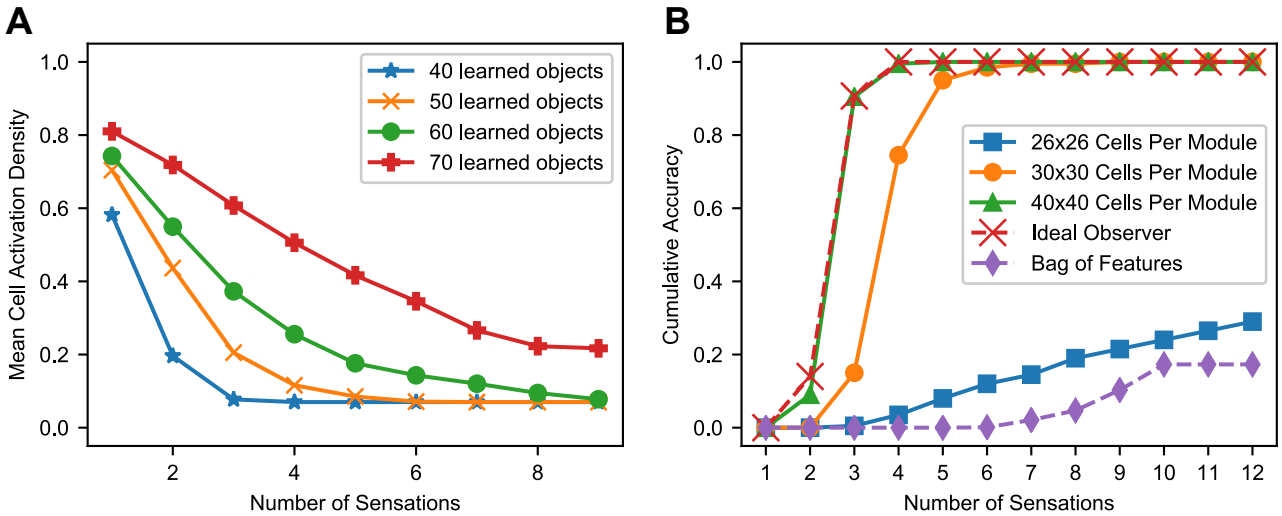

Figure 6. (A) With multiple sensations, the location layer activity converges to a sparse representation. Using the same simulation from Figure 5, we show the activation density after each sensation, averaged across all objects and modules. With additional sensations, the representation becomes sparser until the object is unambiguously recognized. With more learned objects, the network takes longer to disambiguate. The activation density of the first sensation increases with the number of learned objects. If the initial activation density is low, the network converges very quickly. If it's high, convergence can take longer. **(B)** Comparison of this network's performance with the ideal observer and a bag-of-features detector. Each model learned 100 objects from a unique feature pool of size 10. We show the percentage of objects that have been uniquely identified after each sensation, averaged across all objects for ten separate trials. The ideal model compares the sequence of input features and locations to all learned objects while the bag-of-features model ignores locations. With a sufficiently large number of cells per module, the proposed neural algorithm gets similar results to the ideal computational algorithm.

6B, the network with 30x30 cells per module requires more sensations to narrow down the object than the ideal observer, but it always recognizes the object eventually. The network with 26x26 cells per model often never recognizes the object after many sensations, indicating the network has been pushed beyond its capacity.

To understand the way that the system reaches a capacity limit, consider the density of activation in the location modules when a single feature is sensed. If a single feature occurs in many locations, then during learning the location layer and sensory layer reciprocally associate many cells in each module with that feature. Sensing that feature will cause a large percentage of the cells in the location layer to activate. If this percentage is too high, location representations that aren't supposed to be active will be largely contained in this dense activation, resulting in false positives. In the worst case, the location layer will fail to extract anything useful out of this sensory input, because it will activate nearly every location representation as part of its dense activation. We found that the main influence on the model's recognition time is whether the model is approaching its capacity limit. In the following section we characterize this capacity limit.

Capacity varies with size of location layer, statistics of objects. The previous simulations investigated the time to converge onto a unique location. Here we consider the capacity of the network independent of convergence time. We compute the fraction of objects correctly classified after many sensations, and we define the capacity as the maximum number of objects the network can store while maintaining a 90% accuracy rate. While we use 10 locations per object for these simulations, the total number of locations (number of objects times the number of locations per object) is what matters.

The model begins running into capacity limits when the sensory input causes the location layer to activate a union of representations that is large enough to cause false positives in the sensory layer. This occurs when the union contains large portions of location representations that aren't supposed to be active. The likelihood of this event is influenced by the model's number of modules, the number of cells per module, and the statistics of objects. In **Figure 7**, we characterize the impact of these variables on the model's capacity.

The model's capacity increases with the number of modules, though with diminishing returns. In **Figure 7A** we plot two lines to dissect the relationship between the number of modules and capacity. The top line shows the impact of having multiple modules, then the bottom line shows how this effect is reduced by our model's lower dendritic thresholds. Even with a single module, the network is able to converge onto one location representation for a considerable number of learned objects. However, this location representation isn't unique to the object, so it doesn't qualify as recognizing the object; we denote this with a red 'x'. Adding a few more modules ensures unique locations, and each additional module helps the model deal with large unions. They guard the model from having too many false positives when a large percentage of cells are active. More precisely, for each representation, the percentage of the cells that will be active due to randomness will be approximately equal to the activation density, with some variance, and having more modules reduces this variance. If the threshold is 100%, then the model can increase its capacity indefinitely by adding modules.

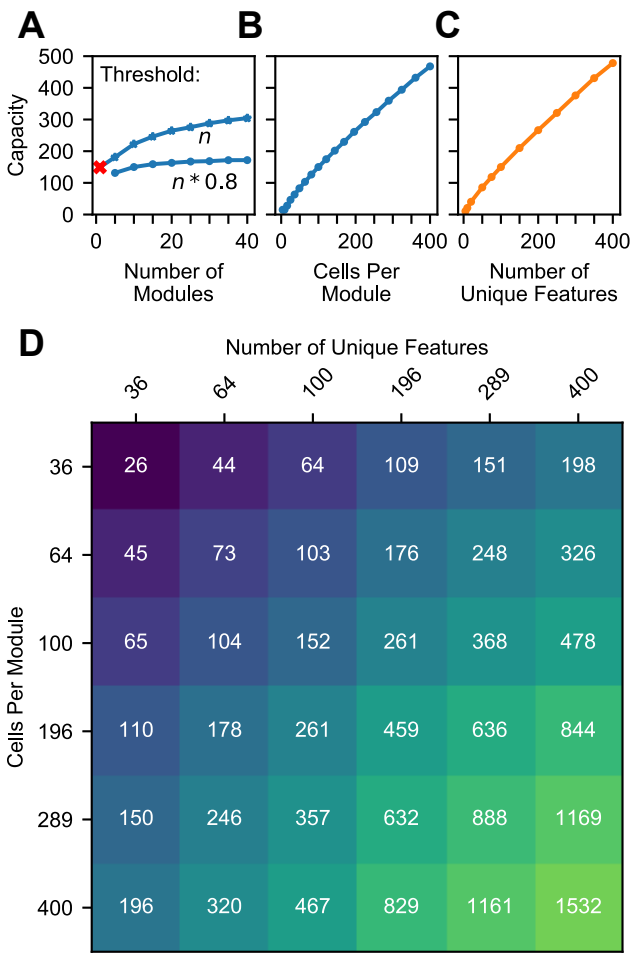

Figure 7. (A-C) Model capacity changes with model parameters (blue) and object parameters (orange). The network's capacity is the maximum number of objects that it can learn while maintaining the ability to uniquely identify at least 90% of the objects. **(A)** Increasing the number of modules in the network increases capacity sublinearly, and this impact depends on the dendritic threshold of the neurons reading this location representation. The two lines show two different thresholds relative to n, the number of modules. With only one module the network can successfully converge to one location, but that location isn't object-specific (red 'x'). **(B)** Increasing the number of cells increases capacity linearly. **(C)** Similarly, increasing the feature pool size also increases capacity linearly. **(D)** A heatmap showing the capacity of the network while varying the feature pool size and location module size**.** An increase in the number of cells per module can compensate for a decrease in the number of unique features.

However, to avoid depending on every neuron reliably firing, this model doesn't use such a high threshold. In our model, neurons will detect a location if 80% of its cells are active. With this lower threshold the benefit of additional modules asymptotes, and no number of modules will be able to handle more than 80% activation density.

Previous analysis of grid cell codes (Fiete et al., 2008) showed that the code's representational capacity increases exponentially with the number of modules. This holds true for this model, but this model's bottleneck is not the size of its unique location spaces. This model's performance depends on its ability to unambiguously represent multiple locations simultaneously. A grid cell code's *union* capacity doesn't scale exponentially with number of modules.

The model's capacity increases linearly with number of cells per module **(Figure 7B)**. As we add additional cells, the size of a bump remains constant relative to the cells, so the bump shrinks relative to the module. Because each bump activates a smaller percentage of the cells in a module, a module can activate more bumps before the network reaches the density at which object classification starts failing.

The model's capacity increases linearly with the number of unique features **(Figure 7C)**. This happens because it reduces the expected total number of occurrences of each feature, and hence the number of elements in each union. This indicates that the statistics of the world influence the capacity of the model, and it also means that this network can improve its capacity by adjusting the "features" that it extracts from sensory input.

Because these two latter parameters have independent linear relationships with capacity, they can compensate for each other. In **Figure 7D** we plot object capacities in a network with 10 modules. We show that increasing (decreasing) the number of cells per module and decreasing (increasing) the number of unique features by the same factor causes the capacity to remain approximately constant. This is illustrated by the approximate symmetry across the chart's diagonal.

The model recognizes an object if the object has at least one sufficiently uncommon feature.

Up to this point, we've characterized this model's ability to recognize objects by stating, "The network will reliably recognize an object if the network has learned fewer than c total objects," and we've measured c. This characterization is built on many assumptions about objects. It's desirable to be able to characterize the model in a way that isn't specific to these assumptions.

Given that the model's ability to recognize objects depends on the density of cell activity invoked by sensory features, we found we could characterize the model's performance more directly by answering, "The network will reliably recognize an object if the object contains a feature with fewer than k total occurrences across all learned objects," and measuring k. In another set of experiments (**Figure S1**), we generated objects using multiple alternate distributions of features. We found that the network's breaking point relative to c (the number of learned objects) did indeed vary widely with the choice of feature distribution, while its breaking point relative to k (the number of locations recalled by sensing a feature) was much more consistent across distributions. This suggests that the network's performance relative to k will hold true with real-world statistics. Using both of these metrics, c and k, we summarize all of these results in **Figure 8**.

We conclude that this model reliably recognizes an object if the object has at least one sufficiently uncommon feature, a feature that causes the network to recall a sufficiently small number of locations. After sensing this feature, the model can use other, more

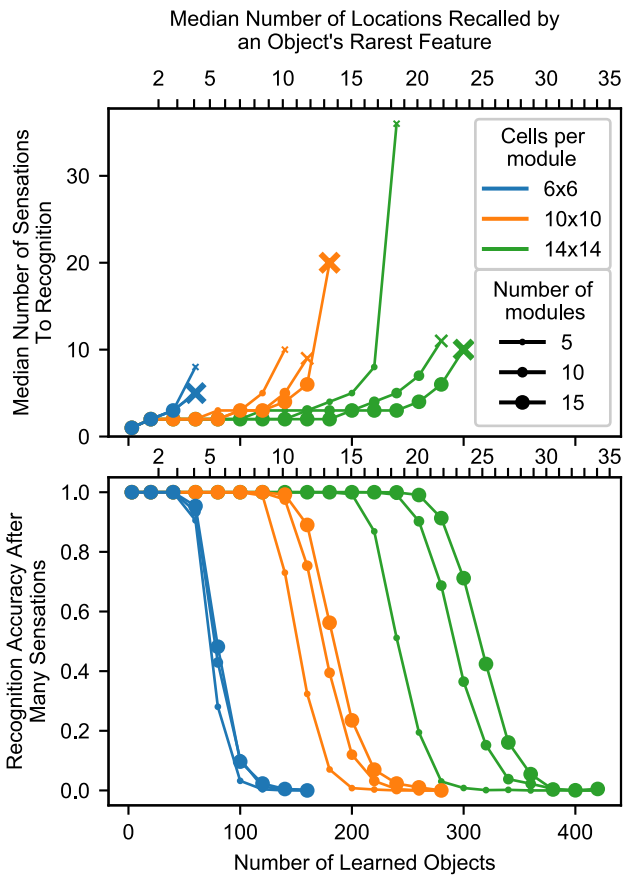

Figure 8. Summary chart showing recognition time and capacity while varying the model parameters. The top chart shows the median number of sensations required to recognize an object, and it marks the final point where a network recognized at least 50% of objects using an 'x'. The bottom chart shows the percentage of objects that were recognized after traversing every point on the object multiple times. The charts share an x axis, and this x axis has two sets of ticks and labels. The bottom ticks are numbers of learned objects, given 10 locations per object and 100 unique features. The top ticks are the typical smallest union sizes that will be invoked by each object. With different statistics, these curves will shift relative to the bottom ticks, but they should stay approximately fixed relative to the top ticks.

common features to finish recognizing the object, but it needs some initial clue to help it activate a manageable union that it can then narrow. With enough cells, this initial clue doesn't need to be especially unique. For example, the feature could have 10 learned locations-on-objects if the module has 10x10 cells per module and 10 modules, as shown in **Figure 8**, and this breaking point can be pushed arbitrarily high by adding more cells. The network can also avoid running into this breaking point by tweaking the set of "features" that it extracts from sensory input.

MAPPING TO BIOLOGY

We have described a two-layer network model for sensorimotor inference and now consider how this network motif maps to known cortical anatomy and physiology. Layer 4 (L4) is well understood to be the primary target of thalamocortical sensory inputs (Douglas and Martin, 2004; Jones, 1998; Theyel et al., 2010). These connections are believed to be driving inputs (Viaene et al., 2011) and target both excitatory and inhibitory neurons, with a slight delay for the inhibitory neurons that provides a window of opportunity for neurons to fire (Harris and Shepherd, 2015). Mountcastle identified the organization of neurons into minicolumns with shared receptive fields (Buxhoeveden, 2002; Mountcastle, 1957).

The connections between L4 and layer 6a (L6a) closely resemble the connections in our model (**Figure 3**, arrows 2 and 4). Thalamic input forms a relatively small percentage of L4 synapses; approximately 45% of its synapses come from L6a (Binzegger et al., 2004). The connections from L6a are weak (Harris and Shepherd, 2015; Kim et al., 2014). They connect to distal dendritic segments of L4 cells whereas thalamocortical afferents connect more proximally (Ahmed et al., 1994; Binzegger et al., 2004). L4 cells also form a significant number of connections to cells in L6a (Binzegger et al., 2004). These biological details closely match our network model, where the location layer (putatively L6a) has a modulatory influence on the sensory input layer (putatively L4) which in turn can drive representations in the location layer.

Our location layer also requires a motor input. Experiments show that L5 cells in motor regions, such as M2, project to sensory regions, including layer 6 (Leinweber et al., 2017; Nelson et al., 2013). There is also a potential indirect pathway through thalamocortical inputs that target layer 6 (Harris and Shepherd, 2015; Thomson, 2010). Thalamic relay cells receive input from layer 5 neurons that are presumed to be efference copies of motor commands sent subcortically (Chevalier and Deniau, 1990; Jones, 1998). Any of these direct or indirect pathways could serve as motor signals for path integration in L6.

Our model draws inspiration from the grid and place cell systems in the hippocampal formation. Our location layer is modeled after grid cells. These cells project (Zhang et al., 2013) to place cells (O'Keefe and Dostrovsky, 1971) in the hippocampus. Areas of hippocampus containing place cells also project back to areas of entorhinal cortex containing grid cells (Rowland et al., 2013). Many place cells seem to represent item-place pairs, and these pairs are learned through experience (Komorowski et al., 2009), a phenomenon that is analogous to the learning of feature-location pairs in our sensory layer.

Location representations in neocortex similar to grid cells are speculative but there is initial experimental support for grid-like codes in neocortex. fMRI experiments with humans performing tasks have led to activity signatures in prefrontal cortex that are similar to grid cell signals (Constantinescu et al., 2016; Julian et al., 2018). Direct cell recordings have also shown grid-like activity in frontal cortex (Jacobs et al., 2013). These experiments are consistent with the hypothesis that grid-like cells are present in the neocortex.

In our model, primary sensory cortex represents the sensory input at locations in an external reference frame. This is consistent with results from (Saleem et al., 2018). As a mouse ran on a virtual track, the majority of recorded cells in primary visual cortex encoded the animal's location on the track, even when the mouse received visual input that occurred at multiple points of the track. According to our

model, in this task the visual cortex represented the location of a sensor (the mouse's eye) relative to an object (the virtual track), and the cortex used the visual input to recall locations on the track while using the mouse's movement to update these location representations.

Although additional experimental work is required, evidence suggests that L4 and L6a provide the best candidate populations for our sensory and location layers, respectively.

DISCUSSION

We have presented a two layer neural network model for sensorimotor object recognition. By processing sensorimotor sequences, it learns objects as spatial arrangements of sensory features. It can recognize learned objects from novel sensorimotor sequences.

The location layer contains modules of cortical grid cells. These cells operate similarly to grid cells in the medial entorhinal cortex. They have motor inputs that shift the activity in each module based on the module's scale and orientation. And they have sensory-derived inputs that activate locations where the input has been previously learned. Path integration updates the current location on the object while sensory-derived inputs narrow down possible locations and correct errors from path integration.

The sensory layer combines location codes with sensory input to create representations of sensory inputs that are unique to objects and locations.

The model provides a concrete implementation of the location signal in our earlier model (Hawkins et al., 2017) but there are a number of practical and theoretical aspects of the model that require further research. There are remaining biological questions about the neural basis for grid cells, extensions for handling orientation, and more. The remainder of this section discusses these topics.

The cell dynamics of grid cell modules

In this model, we treated a grid cell module as a black box with well-known outside properties and unspecified internal neural circuitry, and we simulated the outside properties. We gave this population an additional property that is not typically noted in grid cells: support for unions. The grid cell modules in our model can activate and shift multiple bumps of activity. Various models of grid cell dynamics could be plugged into this model, but this union property introduces a new requirement for these models.

A few different classes of model of grid cell dynamics have been proposed (Giocomo et al., 2011). Recurrent grid cell models have received more empirical support (Yoon et al., 2013) than models in which grid cells establish their responses independent of each other. In recurrent models, grid cells determine their activity using velocity input and connections to other grid cells, either directly or via interneurons. A well-known recurrent model is the continuous attractor network (Burak and Fiete, 2009) which performs robust path integration and offers a simple structural explanation for the origin of the hexagonal firing fields. Another explanation for the origin of these fields is that they are an optimal code for locations which is naturally learned by neural learning rules. This argument has appeared in two lines of research. In path integration models, (Banino et al., 2018) and (Cueva and Wei, 2018) found that recurrent neural networks trained to perform path integration naturally develop grid cells, although neither report the network developing the full rhombus of grid cells at each scale. Setting path integration aside, (Kropff and Treves, 2008), (Dordek et al., 2016), and (Stachenfeld et al., 2017) showed that cells performing Hebbian learning on place cell activity would naturally learn periodic firing fields similar to those of grid cells.

The continuous attractor network is so named because it has a continuous manifold of stable states. If the network activates a representation that isn't within this manifold of stable representations, the activity will move to the nearest stable state. In typical continuous attractor networks, a union is not a stable state, and the network will collapse a union of bumps into a single bump. It's an open question whether it's possible to have a continuous attractor with stable union states. Unions may not be compatible with attractor dynamics. In this work we've shown that it's theoretically appealing for the grid cell module to be able to shift activity around the module without these constrictive attractor dynamics. The attractor dynamics are appealing in part because they explain the hexagonal firing fields, but as mentioned, those may be explainable as the natural result of a recurrent neural network learning a location code.

Modeling this network at a lower level of abstraction is an area for future research.

Representing the orientation of objects

With this model, we showed how cortex could use principles of grid cells to compute the location of a sensor in the reference frame of the sensed object. Notably, this method solved the problem without needing to go through a process of choosing an origin on the object. However, in this formulation, it's still necessary to infer the directions of the axes of the object's reference frame, and this model doesn't yet address this point. Because this model ignores orientations, it will only recognize an object if the object is at its learned orientation relative to the sensor. The model would need to learn objects at every orientation, assigning each a different space of locations.

Just as this model's location representation is inspired by grid cells, an extended version of this model could represent orientation using analogs to head-direction cells (Taube et al., 1990). Grid cells represent the animal's location on a cognitive map, whereas head-direction cells seem to represent the animal's orientation relative to the cognitive map's compass rose. When an animal moves, each entorhinal grid cell module moves its bump of activity according to its direction of movement on the compass rose. Similarly, we expect the neocortex to represent the orientation of sensors in the reference frame of the sensed object, and this orientation will influence how sensor movement translates into the movement of bumps in cortical grid cell modules. Additionally, the sensor's orientation is needed to predict the sensory input. In this extended model, instead of pure location cells projecting to the sensory layer, we expect these cells to represent conjunctive locations and orientations.

Other extensions

This model uses 2D grid cell modules to model 2D objects. The entorhinal cortex's representation of 3D space is an active area of research (Jeffery et al., 2015). This model's general strategy will

work with any 3D location code that provides unique location representations at every 3D location around an object.

This model processes sensory input from a single sensory patch. The body has many sensory patches, and we predict that each of these patches has an instantiation of this network processing its input. In previous work (Hawkins et al., 2017) we showed how these networks work together by using an additional population of cells which represents the current object, invariant to the sensor's location. Each of these networks votes on this classified object. In this way, multiple instantiations of this network work together to recognize objects using the input from multiple independent moving sensors.

As shown in **Figure 7A**, with only one grid cell module this model can still infer an object-centric location, but this location isn't specific to the object, so inferring this location doesn't qualify as object recognition. However, this single-module location would still be useful in an object recognition system. In (Hawkins et al., 2017) we showed that if locations aren't object-specific, the model can still recognize objects if it includes an additional population of cells to represent the current object. This third population receives input from the sensory layer, learning an object as a set of feature-at-location representations. Additionally, if this third population projects back to the sensory layer, it helps the model infer the location.

Relationship to other models

Our model identifies objects using the relative location of sensory features. Objects are disambiguated over time through successive sensations and movements. In contrast, most existing models of object recognition involve a strictly feedforward spatial hierarchical system (DiCarlo et al., 2012; Riesenhuber and Poggio, 1999; Serre et al., 2007; Yau et al., 2009). In these models each level detects the presence of increasingly abstract features in parallel until a complete object is recognized at the top of the hierarchy. Our model implies that each level of a hierarchy might be more powerful than previously assumed. In (Hawkins et al., 2017) we discussed how spatially separated sensory inputs (across multiple cortical columns each computing a location signal) can cooperate in parallel to recognize objects, and some of the implications on hierarchy. Our model suggests a path for integrating sensorimotor behavior into a hierarchical system, and accounts for the many inter and intracortical connections that are not explained by a purely feedforward model. A more detailed study integrating our model into a full hierarchical system is a topic for future research.

There is also a rich history of sensorimotor integration and learning internal models in the context of skilled motor behavior (Wolpert et al., 2011; Wolpert and Ghahramani, 2000). These have primarily focused on learning motor dynamics and kinematic control, such as reaching and grasping tasks. Our model focuses on the more structured object recognition paradigm, but there are many high-level similarities with this body of literature. Our location layer is highly analogous to the forward models posited to exist in motor control (Wolpert et al., 2011). In both cases the current state is updated using a motor efference copy to compute the next state. In both these models, this is an estimate of the state that informs predictions (arrow 2 in **Figure 3**) and is then combined with sensory input to produce the current state (arrows 3 and 4 in **Figure 3**). The primary difference is that our model recognizes a set of structured objects rather than motion trajectories. The neural mechanisms are also significantly different. Nevertheless it is intriguing that the same ideas can be applied to both situations and may reflect a more general design pattern in the brain. An in-depth exploration of this relationship is a topic for future research.

Testable Predictions

Our model makes a number of experimentally testable predictions. We expand on the predictions from (Hawkins et al., 2017).

1. The neocortex uses analogs of grid cell modules to represent locations relative to objects. The cell activity in a module moves through a manifold of representations as the attended location moves relative to an object. For example, in somatosensory areas, cells will respond selectively when the animal's finger is at particular locations relative to an attended object. Just as entorhinal grid cell modules use the same map for every environment, the cells of a single module use the same manifold of representations for every object. This map has limited size, and hence it will perform some form of wrapping at its edges.
2. The neocortex uses a population code of multiple modules to represent object-specific locations.
3. These modules are in Layer 6 of the neocortex.
4. The projection from Layer 6 to Layer 4 modulates which cells in Layer 4 become active. If Layer 6 input is experimentally inhibited, activity in Layer 4 will become denser.
5. The connection from Layer 4 to Layer 6 can drive the Layer 6 cells to become active, but this only occurs when the animal receives an unpredicted input.

MODEL DETAILS

Each module has a fixed number of cells which each have a fixed phase in the rhombus. Gaussian bumps of activity move over these cells. A cell is considered active if its firing rate is sufficiently high. In this section, we walk through the details of these calculations.

Each module contains $w * w$ cells. Each cell c has a constant phase $\vec{\phi}_c$. We partition the 2D range $[0,1) \times [0,1)$ into $w * w$ ranges of equal area and set each cell's $\vec{\phi}_c$ to the center of one of these ranges. Because these modules use basis vectors separated by 60° (Eq. (3)), when mapped onto physical space these cells form a rhombus and they pack together in a hexagonal formation.

The normalized firing rate of a cell c caused by bump b, denoted $r_{c,b}$, is equal to the Gaussian of the distance between them.

$$\text{Gaussian}(d) = e^{-\frac{d^2}{2\sigma^2}} \tag{10}$$

$$r_{c,b} = \text{Gaussian}\left(\text{Distance}\left(\vec{\phi}_c, \vec{\phi}_b\right)\right) \tag{11}$$

$\text{Distance}(\vec{\phi}_c, \vec{\phi}_b)$ represents the shortest distance between the cell and the bump on the phase rhombus. Computing this distance requires changing the basis so that each $\vec{\phi}$ is a point on a rhombus rather than a point on the $[0,1) \times [0,1)$ square. σ specifies the size of the bump relative to the rhombus, which we discuss later.

When there are multiple bumps, the firing rates from each bump are combined as if each rate encodes a probability of an event. The combined firing rate encodes a probability of the "or" of those events.

$$r_c = 1 - \prod_b \left(1 - r_{c,b}\right) \tag{12}$$

To compute module i's active cells $\boldsymbol{A}_t^{\text{loc},i}$, we compute each cell's firing rate and check whether it is above the active firing rate r_{active}.

$$\boldsymbol{A}_t^{loc,i,c} = \begin{cases} 1, & r_c \geq r_{\text{active}} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

We choose r_{active} using a readout resolution parameter $\delta\phi$ which is common in grid cell models (Fiete et al., 2008; Sreenivasan and Fiete, 2011).

$$r_{\text{active}} = \text{Gaussian}\left(\frac{\delta\phi}{2} * \frac{2}{\sqrt{3}}\right) \tag{14}$$

The readout resolution $\delta\phi$ approximately specifies the diameter of the range of phases that a bump encodes. Because modules are 2D, if the readout resolution is 1/4 then the bump can encode approximately 16 possible positions in the rhombus. The multiplicative factor of $2/\sqrt{3}$ accounts for the fact that when circles pack together in hexagonal formation, they leave some area uncovered; this factor expands the circles to overlap and cover this area. With a single bump, changing the σ parameter has no effect on the model, because the $\delta\phi$ parameter has complete control over what fraction of the cells are considered active. σ becomes relevant when there are multiple bumps. The wider these bumps are, the more they'll combine and cause interstitial cells' firing rates to rise above r_{active}.

We vary σ and $\delta\phi$ as follows. Our baseline parameters mimic the sparsity of rat entorhinal grid cell modules. We set σ to 0.18172, a number we obtained by fitting a 2D Gaussian to the firing fields generated by the model in (Monaco and Abbott, 2011). We set $\delta\phi$ to a conservative estimate of 1/3. In this configuration, we assign the network 6x6 cells, and a bump always activates at least 2x2 cells. When we test the network with more cells, we assume that the bump remains a fixed size relative to the cells, i.e. that the bump is smaller relative to the size of the module, and we scale down σ and $\delta\phi$ accordingly. For example, with 12x12 cells, we use $\sigma/2$ and $\delta\phi\ /\ 2$. By varying the parameters in this way, a single bump always activates between 4 and 7 cells, depending on where the bump is centered relative its local neighborhood of cells, and as we vary the number of cells we're effectively varying the number of cells that the bump *doesn't* activate.

During learning, only the cell with the highest firing rate is associated with the sensed feature. (When a sensed feature activates a cell, it activates a bump centered on the cell via Eq. (7), activating the cells around it.) This means this model's learning resolution is twice as precise as the readout resolution. Because sensed features are associated with cells that represent a range of phases, there's always some uncertainty in the phase recalled by sensory input. If the learned resolution weren't more precise than the readout resolution, the bump of active cells would need to expand to account for this uncertainty, and the effective readout resolution would be half as precise. Using fixed-sized bumps, achieving a particular readout resolution – that is, having a bump encode a range of phases with a particular diameter – requires the learning resolution to be at least twice as precise as this readout resolution.

The ideal classifier in **Figure 6B** stores all objects as 2D arrays. During inference, it uses the first sensed feature to find all possible locations on all objects with that feature, and it stores these as candidate locations. With each subsequent movement it updates all of the candidate locations. Any updated candidates that are not valid locations on objects or contain features that don't match the new sensed feature are removed from the candidate list. Once there is only a single location left, inference is successfully completed.

The bag of features detector stores a set of features for each learned object. It does not keep track of how many times features occur, just the set of unique features present somewhere on the object. During inference, another set keeps track of which features have been sensed so far. Once there is only one object that contains all of the sensed features, inference is successfully completed. If there are multiple objects that contain all features once all locations on the object being tested have been visited, then the object cannot be uniquely classified.

Code availability

All of the source code for this model and these simulations can be found at https://github.com/numenta/htmpapers.

REFERENCES

Ahmed, B., Anderson, J. C., Douglas, R. J., Martin, K. A. C., and Nelson, J. C. (1994). Polyneuronal innervation of spiny stellate neurons in cat visual cortex. *J. Comp. Neurol.* 341, 39–49. doi:10.1002/cne.903410105.

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., et al. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature* 557, 429–433. doi:10.1038/s41586-018-0102-6.

Barry, C., Hayman, R., Burgess, N., and Jeffery, K. J. (2007). Experience-dependent rescaling of entorhinal grids. *Nat. Neurosci.* 10, 682–684. doi:10.1038/nn1905.

Binzegger, T., Douglas, R. J., and Martin, K. A. C. (2004). A Quantitative Map of the Circuit of Cat Primary Visual Cortex. *J. Neurosci.* 24, 8441–8453. doi:10.1523/JNEUROSCI.1400-04.2004.

Burak, Y., and Fiete, I. R. (2009). Accurate path integration in continuous attractor network models of grid cells. *PLoS Comput. Biol.* 5. doi:10.1371/journal.pcbi.1000291.

Buxhoeveden, D. P. (2002). The minicolumn hypothesis in neuroscience. *Brain* 125, 935–951. doi:10.1093/brain/awf110.

Chevalier, G., and Deniau, J. M. (1990). Disinhibition as a basic process in the expression of striatal functions. *Trends Neurosci.* 13, 277–280. doi:10.1016/0166-2236(90)90109-N.

Constantinescu, A. O., OReilly, J. X., and Behrens, T. E. J. (2016). Organizing conceptual knowledge in humans with a gridlike code. *Science (80-.).* 352, 1464–1468. doi:10.1126/science.aaf0941.

Cueva, C. J., and Wei, X.-X. (2018). Emergence of grid-like representations by training recurrent neural networks to

perform spatial localization. 1–15.
DiCarlo, J. J., Zoccolan, D., and Rust, N. C. (2012). How Does the Brain Solve Visual Object Recognition? *Neuron* 73, 415–434. doi:10.1016/J.NEURON.2012.01.010.
Dordek, Y., Soudry, D., Meir, R., and Derdikman, D. (2016). Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis. *Elife* 5, 1–36. doi:10.7554/eLife.10094.
Douglas, R. J., and Martin, K. A. C. (2004). Neuronal Circuits of the Neocortex. *Annu. Rev. Neurosci.* 27, 419–451. doi:10.1146/annurev.neuro.27.070203.144152.
Fiete, I. R., Burak, Y., and Brookings, T. (2008). What Grid Cells Convey about Rat Location. *J. Neurosci.* 28, 6858–6871. doi:10.1523/JNEUROSCI.5684-07.2008.
Giocomo, L. M., Moser, M. B., and Moser, E. I. (2011). Computational models of grid cells. *Neuron* 71, 589–603. doi:10.1016/j.neuron.2011.07.023.
Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., and Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature* 436, 801–806. doi:10.1038/nature03721.
Harris, K. D., and Shepherd, G. M. G. (2015). The neocortical circuit: themes and variations. *Nat. Neurosci.* 18, 170–181. doi:10.1038/nn.3917.
Hawkins, J., and Ahmad, S. (2016). Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. *Front. Neural Circuits* 10, 1–13. doi:10.3389/fncir.2016.00023.
Hawkins, J., Ahmad, S., and Cui, Y. (2017). A Theory of How Columns in the Neocortex Enable Learning the Structure of the World. *Front. Neural Circuits* 11, 81. doi:10.3389/FNCIR.2017.00081.
Jacobs, J., Weidemann, C. T., Miller, J. F., Solway, A., Burke, J. F., Wei, X. X., et al. (2013). Direct recordings of grid-like neuronal activity in human spatial navigation. *Nat. Neurosci.* 16, 1188–1190. doi:10.1038/nn.3466.
Jeffery, K. J., Wilson, J. J., Casali, G., and Hayman, R. M. (2015). Neural encoding of large-scale three-dimensional space—properties and constraints. *Front. Psychol.* 6, 1–12. doi:10.3389/fpsyg.2015.00927.
Jones, E. G. (1998). Viewpoint: The core and matrix of thalamic organization. *Neuroscience* 85, 331–345. doi:10.1016/S0306-4522(97)00581-2.
Julian, J. B., Keinath, A. T., Frazzetta, G., and Epstein, R. A. (2018). Human entorhinal cortex represents visual space using a boundary-anchored grid. *Nat. Neurosci.* 21, 191–194. doi:10.1038/s41593-017-0049-1.
Kim, J., Matney, C. J., Blankenship, A., Hestrin, S., and Brown, S. P. (2014). Layer 6 corticothalamic neurons activate a cortical output layer, layer 5a. *J Neurosci* 34, 9656–9664. doi:10.1523/JNEUROSCI.1325-14.2014.
Komorowski, R. W., Manns, J. R., and Eichenbaum, H. (2009). Robust Conjunctive Item-Place Coding by Hippocampal Neurons Parallels Learning What Happens Where. *J. Neurosci.* 29, 9918–9929. doi:10.1523/JNEUROSCI.1378-09.2009.
Kropff, E., and Treves, A. (2008). The emergence of grid cells: Intelligent design or just adaptation? *Hippocampus* 18, 1256–1269. doi:10.1002/hipo.20520.
Leinweber, M., Ward, D. R., Sobczak, J. M., Attinger, A., and Keller, G. B. (2017). A Sensorimotor Circuit in Mouse Cortex for Visual Flow Predictions. *Neuron* 95, 1420–1432.e5. doi:10.1016/j.neuron.2017.08.036.
Major, G., Larkum, M. E., and Schiller, J. (2013). Active properties of neocortical pyramidal neuron dendrites. *Annu. Rev. Neurosci.* 36, 1–24. doi:10.1146/annurev-neuro-062111-150343.
Marr, D., and Nishihara, H. K. (1978). Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes. *Proc. R. Soc. B Biol. Sci.* 200, 269–294. doi:10.1098/rspb.1978.0020.
McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., and Moser, M.-B. (2006). Path integration and the neural basis of the "cognitive map." *Nat. Rev. Neurosci.* 7, 663–678. doi:10.1038/nrn1932.
Monaco, J. D., and Abbott, L. F. (2011). Modular Realignment of Entorhinal Grid Cell Activity as a Basis for Hippocampal Remapping. *J. Neurosci.* 31, 9414–9425. doi:10.1523/JNEUROSCI.1433-11.2011.
Mountcastle, V. B. (1957). MODALITY AND TOPOGRAPHIC PROPERTIES OF SINGLE NEURONS OF CAT'S SOMATIC SENSORY CORTEX. *J. Neurophysiol.* 20, 408–434. doi:10.1152/jn.1957.20.4.408.
Nelson, A., Schneider, D. M., Takatoh, J., Sakurai, K., Wang, F., and Mooney, R. (2013). A Circuit for Motor Cortical Modulation of Auditory Cortical Activity. *J. Neurosci.* 33, 14342–14353. doi:10.1523/JNEUROSCI.2275-13.2013.
O'Keefe, J., and Dostrovsky, J. (1971). The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain Res.* 34, 171–175. doi:10.1016/0006-8993(71)90358-1.
Ocko, S., Hardcastle, K., Giocomo, L. M., and Ganguli, S. (2018). Emergent Elasticity in the Neural Code for Space. *bioRxiv*, 326793. doi:10.1101/326793.
Riesenhuber, M., and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nat. Neurosci.* 2, 1019–25. doi:10.1038/14819.
Rowland, D. C., Weible, A. P., Wickersham, I. R., Wu, H., Mayford, M., Witter, M. P., et al. (2013). Transgenically Targeted Rabies Virus Demonstrates a Major Monosynaptic Projection from Hippocampal Area CA2 to Medial Entorhinal Layer II Neurons. *J. Neurosci.* 33, 14889–14898. doi:10.1523/JNEUROSCI.1046-13.2013.
Saleem, A. B., Diamanti, E. M., Fournier, J., Harris, K. D., and Carandini, M. (2018). Coherent encoding of subjective spatial position in visual cortex and hippocampus. *Nature*. doi:10.1038/s41586-018-0516-1.
Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., and Poggio, T. (2007). Robust Object Recognition with Cortex-Like Mechanisms. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 411–426. doi:10.1109/TPAMI.2007.56.
Sreenivasan, S., and Fiete, I. (2011). Grid cells generate an analog error-correcting code for singularly precise neural computation. *Nat. Neurosci.* 14, 1330–1337. doi:10.1038/nn.2901.
Stachenfeld, K. L., Botvinick, M. M., and Gershman, S. J. (2017). The hippocampus as a predictive map. *Nat. Neurosci.* 20, 1643. Available at: http://dx.doi.org/10.1038/nn.4650.
Taube, J. S., Muller, R. U., and Ranck, J. B. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis. *J. Neurosci.* 10, 420–35. doi:10.1212/01.wnl.0000299117.48935.2e.
Theyel, B. B., Llano, D. A., and Sherman, S. M. (2010). The corticothalamocortical circuit drives higher-order cortex in

the mouse. *Nat. Neurosci.* 13, 84–88. doi:10.1038/nn.2449.The.
Thomson, A. M. (2010). Neocortical layer 6, a review. *Front. Neuroanat.* 4, 1–14. doi:10.3389/fnana.2010.00013.
Viaene, A. N., Petrof, I., and Sherman, S. M. (2011). Synaptic Properties of Thalamic Input to the Subgranular Layers of Primary Somatosensory and Auditory Cortices in the Mouse. *J. Neurosci.* 31, 12738–12747. doi:10.1523/JNEUROSCI.1565-11.2011.
Wolpert, D. M., Diedrichsen, J., and Flanagan, J. R. (2011). Principles of sensorimotor learning. *Nat. Rev. Neurosci.* 12, 739–51. doi:10.1038/nrn3112.
Wolpert, D. M., and Ghahramani, Z. (2000). Computational principles of movement neuroscience. *Nat. Neurosci.* 3, 1212–1217. doi:10.1038/81497.
Yau, J. M., Pasupathy, A., Fitzgerald, P. J., Hsiao, S. S., and Connor, C. E. (2009). Analogous intermediate shape coding in vision and touch. *Proc. Natl. Acad. Sci. U. S. A.* 106, 16457–16462. doi:10.1073/pnas.0904186106.
Yoon, K., Buice, M. A., Barry, C., Hayman, R., Burgess, N., and Fiete, I. R. (2013). Specific evidence of low-dimensional continuous attractor dynamics in grid cells. *Nat. Neurosci.* 16, 1077–1084. doi:10.1038/nn.3450.
Zhang, S. J., Ye, J., Miao, C., Tsao, A., Cerniauskas, I., Ledergerber, D., et al. (2013). Optogenetic dissection of entorhinal-hippocampal functional connectivity. *Science (80-.).* 340, 44. doi:10.1126/science.1232627.

Supplementary figures

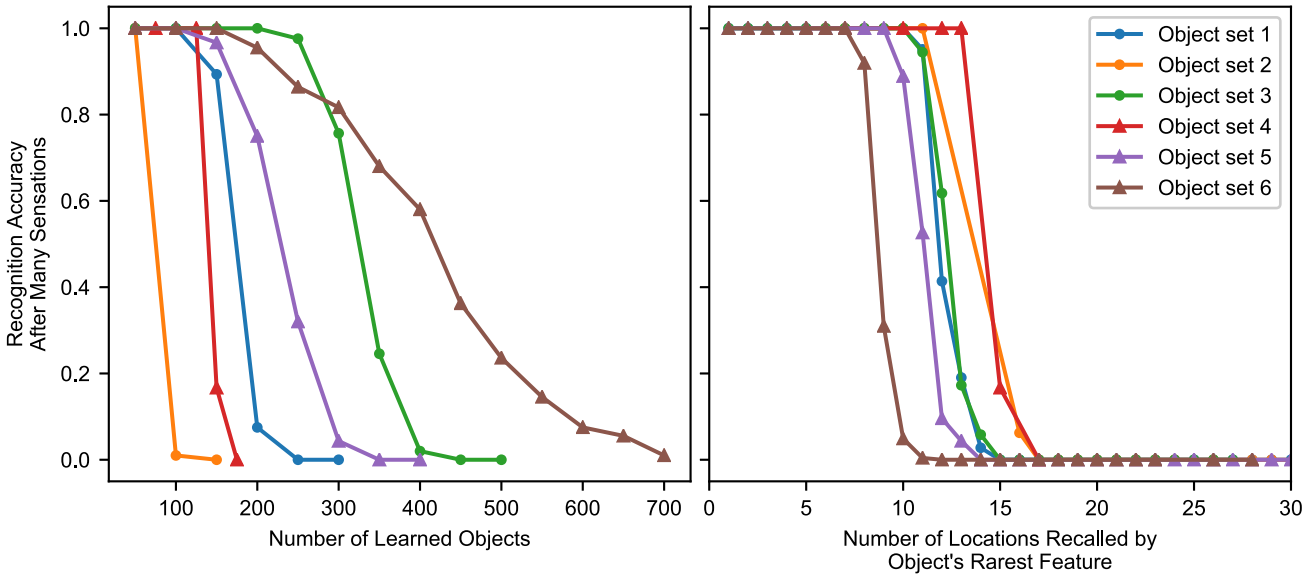

Figure S1. Varying the object statistics, the model's breaking point varies significantly relative to number of learned objects. The breaking point is much more consistent relative to the number of locations recalled by object features. In these charts we use a single model and test on 6 different distributions of objects. The model uses 10 modules with 10x10 cells per module. **(Left)** The network's capacity depends on the statistics of objects. The network's performance begins to break down after a certain number of objects, and this breaking point can vary by orders of magnitude with different object distributions. **(Right)** This breaking point varies significantly less when described in terms of "number of locations recalled by a sensation" rather than "number of objects learned". Using the same data from the first chart, for each object we measure the total number of occurrences of the object's rarest feature, and we plot recognition accuracy against this number. With each of these object distributions, the model reaches its breaking point when the number of recalled locations is within a small interval – conservatively, between 7 and 15. There is still some variation due to the statistics of the object's other features (not just its rarest feature), but the number of occurrences of the rarest feature provides a good first approximation for whether the network will recognize the object. **(Object descriptions)** Each object set had 100 unique features and 10 features per object, except where otherwise noted. The first three sets generate objects using the same strategy as all the other simulations, varying the parameters. The last three use different strategies. *Object Set 1:* baseline. *Object Set 2:* 40 unique features rather than 100. *Object Set 3:* 5 features per object rather than 10. *Object Set 4:* Every feature occurs the same number of times, +/- 1, rather than each object being randomly selected set of features with replacement. *Object Set 5:* Bimodal distribution of features, probabilistic. Divide features into two equal-sized pools, choose features from the second pool more often than features from the first. *Object Set 6:* Bimodal distribution of features, enforced structure. The features are divided equally into pools. Each object consists of one feature from the first pool and nine from the second.