Constructing future behaviour in the hippocampal formation through composition and replay

Jacob J.W. Bakermans[1], Joseph Warren[1,4], James C.R. Whittington[1,2*], Timothy E.J. Behrens[1,3,4*]

Hippocampus is critical for memory, imagination, and constructive reasoning. However, recent models have suggested that its neuronal responses can be well explained by state-spaces that model the transitions between experiences. How do we reconcile these two views? Here we show that if state-spaces are constructed compositionally from existing primitives, hippocampal responses can be interpreted as compositional memories, binding these primitives together. Critically, this enables agents to behave optimally in novel environments with no new learning, inferring behaviour directly from the composition. This provides natural interpretations of generalisation and latent learning. Hippocampal replay can build and consolidate these compositional memories, but importantly, due to their compositional nature, it can construct states it has never experienced - effectively building memories of the future. This enables new predictions of optimal replays for novel environments, or after structural changes. Together, these findings provide a framework for reasoning about several seemingly disparate functions of hippocampus.

A recent spate of hippocampal models suggest that hippocampus represents a state-space and its transitions, i.e. a cognitive map (George et al., 2021; Piray & Daw, 2021; Stachenfeld et al., 2017; Whittington et al., 2020). These models come in two flavours, those that infer the state-space from sequences (George et al., 2021; Whittington et al., 2020), and those that use the state-space for reinforcement learning (RL) (Piray & Daw, 2021; Stachenfeld et al., 2017). Together they explain many key hippocampal findings; from associative learning in neuroimaging studies (Garvert et al., 2017; Schapiro et al., 2016) to precise cellular responses during spatial sequences, such as place (O'Keefe, 1976) and grid cells (Hafting et al., 2005). Additionally, they account for latent state representations in reinforcement learning tasks that require non-spatial behaviour, such as splitter cells (Frank et al., 2000; Wood et al., 2000) in spatial alternation tasks or lap cells (Sun et al., 2020) in tasks that require counting (George et al., 2021; Whittington et al., 2020). Overall, these models' successes have been greatly suggestive that hippocampus builds state-spaces from sequences and may use these state-spaces for RL.

However, these new ideas about state-space inference seem at odds with key principles of hippocampal function that are supported by a wealth of empirical evidence. Most strikingly, hippocampus' primary role is one of memory (Eichenbaum & Cohen, 2014; Scoville & Milner, 1957). This memory is part of a constructive process that also supports imagination, and scene construction and understanding (Addis et al., 2007; Hassabis & Maguire, 2007; Mullally et al., 2012; Rosenbaum et al., 2009; Tulving, 1985). This evidence suggests that the hippocampal representation is compositional, binding cortical information together to build a representation of the current experience. A door to the north-west. A wall to the south. A friend sitting at the table. Indeed, hippocampal neurons respond to conjunctions of external features (Komorowski et al., 2009), as if binding together elements of a composition, and RL state-space tasks predominantly rely on hippocampus when new state-spaces are initially constructed (Packard & McGaugh, 1996). The big computational benefit to compositional scene-construction and episodic memories is being able to understand and respond to situations in *one-shot*. This flexibility is missing from the traditional state-space models: learning the state-space often requires much experience, and can be brittle to policy or local transition changes (Russek et al., 2017). While these symptoms can be alleviated, for example by offline replay that performs credit assignment when new rewards or barriers are observed (Mattar & Daw, 2018; Sutton & Barto, 2018), the diagnosis remains the same. This raises a significant puzzle - how do the state-space models relate to the well-known memory and construction machinery of hippocampus?

1. Wellcome Centre for Integrative Neuroimaging, University of Oxford, Oxford, UK. 2. Department of Applied Physics, Stanford University, Stanford, CA, USA. 3. Wellcome Centre for Human Neuroimaging, University College London, London, UK. 4. Sainsbury Wellcome Centre for Neural Circuits and Behaviour, University College London, London, UK; *contributed equally

Here, we unify the two through a model of *state-space composition*. To be compositional, the model of the world must be decomposable into representational sub-blocks ($z = [z^1, z^2, z^3, \dots]$), where the dynamics of each sub-block are independent from one another (${z^i}_t = g({z^i}_{t-1})$). This means that new configurations of the sub-blocks (corresponding to a new world model) will have predictable dynamics with no extra learning. In this scenario, the world-model for any particular situation specifies how the building blocks combine in the current world - a role we propose for hippocampal place cells[1]. For example, composing a map of space with an object centric map (or wall- or door-centric), means the hippocampal state-space knows where the object (or wall or door) currently is in space. This feature of hippocampal compositions offers several advantages over and above hippocampus being just state-spaces or memories alone, and provides new insights into hippocampal phenomena. We show that 1) Conjunctive hippocampal cells can be reinterpreted as compositionally binding together multiple maps/variables, accounting for a variety of hippocampal cells and predicting experimentally observed non-random remapping; 2) There is a dramatic performance gain (versus standard RL) when hippocampal state-spaces are compositions of *already learned* building blocks, since policies learned on one hippocampal composition generalise to novel compositions; 3) Latent learning can be understood as building compositions in the absence of reward, so that optimal behaviour can be achieved when rewards are discovered; 4) Replay can compose states-spaces offline into memories that improve future behaviour either by updating policies or consolidating existing memories; 5) We can predict optimal replay patterns.

RESULTS

Model of hippocampal compositions

We propose that hippocampus makes use of reusable building blocks that it can compose together to understand new situations - just like scene construction (Fig 1a). For clarity, we explain this model in terms of spatial representations, but it applies to any situation where the world-dynamics can be split into compositional parts. In this section, we give a high-level overview of the model and its main features, which we will further elaborate on in each of the following sections. The Methods section provides further implementational details.

In space, to construct a scene of a simple room, hippocampus may bind together a representation of where you are in space, x, with where you are relative to walls (w), salient objects (o), and rewards (r) (Fig 1b). This means that hippocampal cells will be a conjunction of representations for space, walls, objects, and rewards (x, w, o, r). Critically, the subcomponents (x, w, o, r) are reusable since any room is a different configuration of space, walls, objects, and rewards. Thus, the hippocampal state-space can be built immediately, for any new environment, out of different compositions of these building blocks (Fig 1c).

[1] We refer to the cortical building blocks as *compositional* as they can occur in any configuration, but we propose the *composition* of these building blocks occurs in hippocampus.

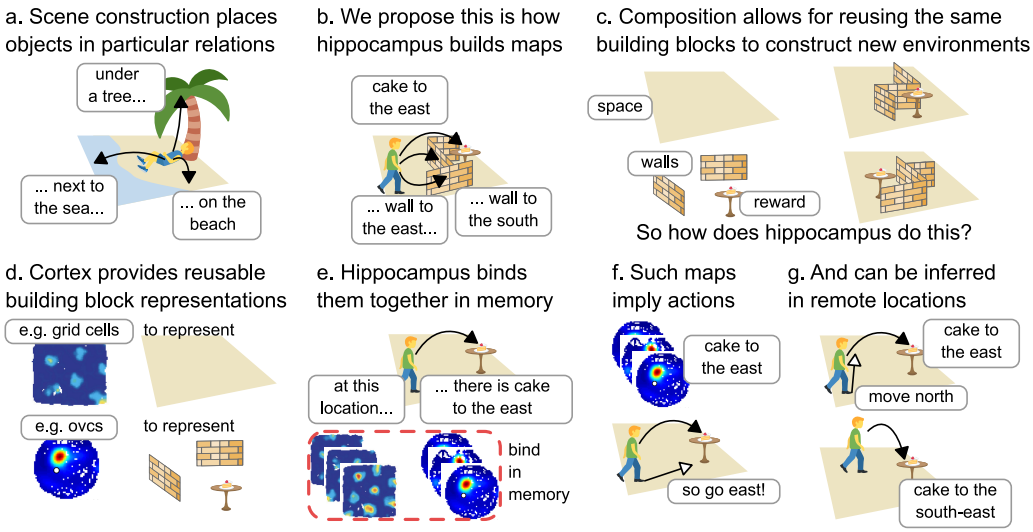

Fig 1. Model of hippocampal state-space composition. a. To construct a scene, like this (imagined) experience of being on the beach, hippocampus binds objects into relational configurations. **b**. We propose that hippocampus similarly composes state-spaces from structural elements like walls and rewards (black arrowheads denote allocentric vector relations). **c**. The advantage of composition is that new situations, like new spatial environments, can be constructed from the same building blocks. **d**. Cortex can provide such reusable representations, for example in the form of grid cells (Hafting et al., 2005) and object vector cells (OVCs; (Høydal et al., 2019)) in spatial environments. **e**. Hippocampus can construct new situations from cortical representations by binding them together in relational memory. **f**. Because these compositional state-spaces are built from reusable elements, understanding from one environment generalises to others. In particular, these maps immediately imply actions (white arrowheads). **g**. To construct the state-space, building block representations can be propagated to remote locations - online, but also offline in replay.

A key question, however, is what should the building blocks look like? Fortunately, for space, walls, objects, and rewards (x, w, o, r), biology already tells us. In entorhinal cortex and hippocampus, along with place and grid cells that code for space, there are vector cells that point towards walls, objects, and rewards: border-vector cells, object-vector cells, reward-vector cells (Gauthier & Tank, 2018; Høydal et al., 2019; Lever et al., 2009; Solstad et al., 2008). Each cell provides a distance and direction to the border/object/reward, and each population of vector cells for border/object/reward provides a map that can be path integrated (updated with respect to actions taken), just like grid cells for space, but rather with each map being centred around the border/object/reward (Fig 1d). In other words, each population of cell types represents a coordinate system, e.g. grid cells are a global 2D coordinate system, object vector cells are also a 2D coordinate system but locally centred on objects, etc. We posit non-spatial building blocks will have coordinate system representations too, i.e., vector cells but in non-spatial coordinates (Nieh et al., 2021). Lastly, these cells generalise and so are reusable - an object vector cell in one environment is also an object vector cell in another environment, with the same true for grid cells and border vector cells.

The reusability of these building blocks means any understanding from one configuration can be generalised to new configurations. This is particularly powerful for reinforcement learning, as now an agent does not have to learn a new policy from scratch for every new environment (like RL on conventional hippocampal state-spaces, e.g. SR (George et al., 2021; Stachenfeld et al., 2017)). Instead, the compositional state-space already implies actions (Fig 1f). A reward vector cell says head towards the reward, a border vector cell says don't crash into the border and so on. In RL terminology, credit assigned to these compositional building blocks in one situation is a useful assignment in new situations as well, i.e. the hippocampal state-space comes 'pre-credit assigned'. This means online RL is dramatically reduced and often not necessary at all.

But how can this composition be achieved in the first place? It requires binding representations of (x, w, o, r) at every location in the environment and in the appropriate configuration, i.e. binding the vector representation saying 3 steps north to the reward to the spatial representations when you are actually 3 steps south of the reward. This is easily achieved by binding the reward vector representation, r, at the current timestep to the spatial location representation, x, at the current timestep and storing it as a hippocampal memory (Fig 1e). However, performing compositions in online behaviour is slow to propagate information throughout the whole state-space, and is error prone as it relies on path integration (Etienne & Jeffery, 2004; Mittelstaedt & Mittelstaedt, 1980). Fortunately, biology provides a potential resolution in the form of offline replay (Foster & Wilson, 2006), which can bind building blocks together in remote locations (Fig 1g). This is both more data efficient and reduces potential errors. In this context, replay is effectively performing credit assignment since it constructs the state-space for future successful behaviour; next time we are at that remote location we already know about the reward. This function of replay also has implications for what the content of replay should be: the best replays are those that improve the map the most. We can therefore predict patterns of replay and behaviour from this principle.

Place cells that embed global knowledge in local representations

The constructive nature of hippocampal function suggests a particular interpretation of hippocampal place cells - a conjunctive representation (Komorowski et al., 2009; Manns & Eichenbaum, 2006), where hippocampal cells bind existing representations into a new relational configuration. At its simplest this means combining sensory input with the grid coordinate system, as suggested by previous sequence models (Whittington et al., 2020), but much more is required for it to be a useful state-space for inducing behaviour. The local (at a given location) representation must contain global (about other locations) relational knowledge. If a reward is 4 steps east, but there is a wall 3 steps east, you should move north or south (Fig 2a). Such a representation can also be built from conjunctions, but they must be conjunctions between cortical cells that encode this global relational knowledge, such as object- and border-vector cells in the medial entorhinal cortex (Fig 2b). With such a representation, unlike with existing place cell models, many transitions are inherited, rather than learnt, as they are implied by the particular combination of cortical inputs (Fig 2c).

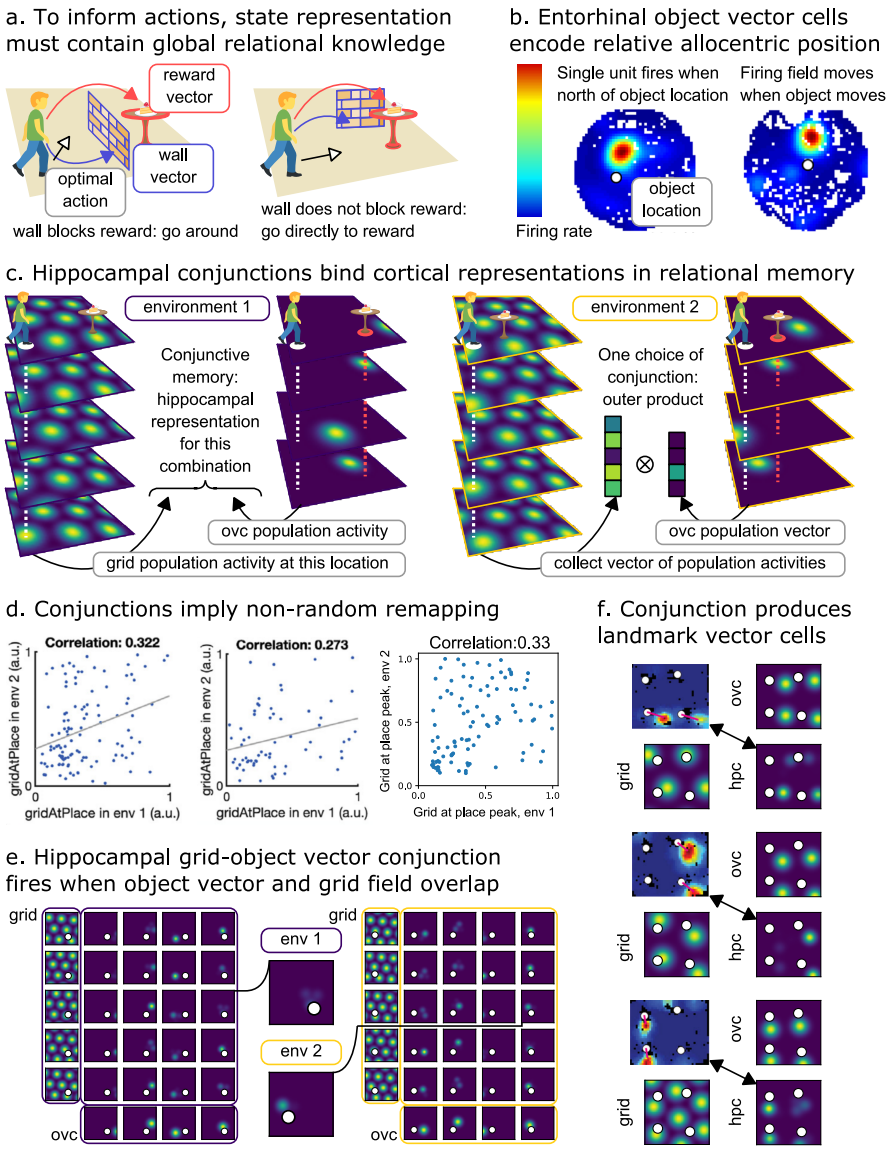

Fig 2. Hippocampal place cells as conjunction of cortical building blocks. a. A state representation that induces behaviour needs to contain global relational information about the structural elements of the environment. If there is a wall to the east and a reward further to the east, the correct course of action is to go north around the wall, but if the wall is to the north, the agent can proceed directly towards the reward. **b**. Populations of object vector cells (Høydal et al., 2019), each of which fires at a particular allocentric angle and distance from an object, encode this kind of global relational knowledge. **c**. Hippocampus combines cortical building block representations, like the population of 5 grid cells for space and 4 object-vector cells for reward, into a new conjunctive memory. One way to achieve this conjunctive representation is by encoding the outer product of the population vectors in hippocampal memory. **d.** Conjunctive hippocampal cells remap between environments, but do so non-randomly: they still respond to the same cortical inputs. Place cells, modelled as location-observation conjunctions, appear at similar grid phases in different environments, replicating (right panel) the results from (Whittington et al., 2020) (left and middle panel). **e**. For the conjunction of grid and object-vector codes this remapping means that a hippocampal neuron may be active in environment 1 but not in environment 2, depending on whether the object-vector and grid peaks align. **f**. The same mechanism can produce hippocampal landmark-vector cells (empirical: top left (Deshmukh & Knierim, 2013), simulated: bottom right) that respond to some objects in the environment but not to others.

When examined in an open-field arena, it is not possible to distinguish this representation from simple place cells, or states in a state-space model, but compositional representations should have particular remapping

properties when the environment changes, or when objects appear in multiple places in the environment (Methods 3. Conjunctive memories). For example if one compositional input is grid cells, then when place cells remap (Anderson & Jeffery, 2003; Bostock et al., 1991; Muller & Kubie, 1987) they should remap to the same grid phase in each environment (Fig 2d), as shown in (Whittington et al., 2020). If another compositional input is object vector cells, hippocampal cells should appear at the same vector from multiple objects, but only if that coincides with the same grid phase (Fig 2e) - a possible explanation of landmark vector cells (Fig 2f; (Deshmukh & Knierim, 2013)).

Notably, such conjunctive representations need not be the only representations in hippocampus, and can be further tuned by learning as the environment becomes familiar, or behaviour is overlearned (for example, with an algorithm that builds a successor representation). Their power comes from their potential to generate behaviour immediately in a new environment.

Compositional codes facilitate zero-shot behaviour

To explore this potential for zero-shot generalisation we compare two agents (Fig 3a). In both cases, we train a feedforward network f through supervised learning on ground-truth optimal policies to predict optimal actions a given a state representation s**.** In the first agent (Traditional) states are unrelated to the features across environments: they only represent space x. In the second agent (Compositional), states are compositions of vector cells to environment features: they combine walls, objects, and rewards (w, o, r). We do not include space (x) as part of this composition as it is not required for action selection (but we will need the spatial component later to build compositional maps). In each case, we train on multiple environments with walls, objects, and rewards, placed at random. The optimal policy is defined as the local actions that minimise the number of steps to reward from each location.

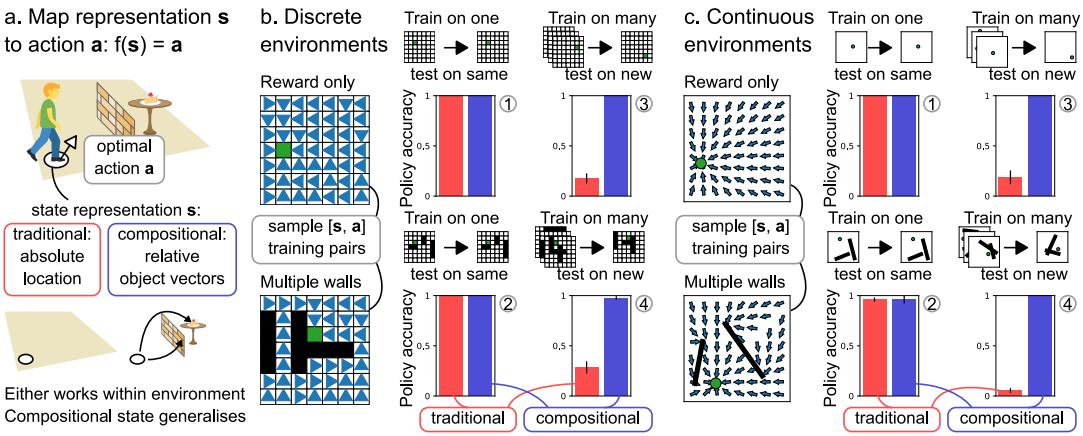

Fig 3. Compositional state representations generalise. a. We learn a mapping f from state representation s to optimal action a (white arrowhead): $f(s) = a$. For a given location, s represents the absolute location in the environment (traditional; red; $s = [x]$) or the relative vector codes (black arrowheads) for all objects (walls and rewards) in the environment (compositional; blue; $s = [w, r]$). **b**. In discrete graph environments, we sample [state representation, optimal action (blue triangles)] training examples, both in simple environments with reward only and in complex environments with multiple walls. When trained on a single environment and tested on the same environment, both the traditional and compositional state representations provide accurate policies (panels 1 and 2). Only mappings from compositional state representations yield accurate policies when tested in a new environment (panels 3 and 4). **c**. In continuous environments, where locations are continuous coordinates and actions are continuous directions, we find the same results. Either representation works within environments (panels 1 and 2), but only the compositional state representation generalises (panels 3 and 4).

In each simulation, we randomly generate a set of environments and calculate optimal policies. We then sample [state representation, optimal action] pairs (s, a) as training examples to train a feedforward network that maps state representations to optimal actions through supervised learning. We evaluate the network's performance by measuring whether following the learned policy successfully navigates to the goal from a set of test locations (Methods 4. Policies that generalise).

While the traditional agent is able to learn arbitrarily complex policies in a single environment with the reward in a fixed location (Fig 3b1, 3b2), it immediately fails when either the reward or environmental features change (Fig 3b3, 3b4). This is unsurprising as it is not re-trained when the environment changes, and the state representation does not carry useful information across environments. By contrast, the compositional agent immediately generalises behaviour. Given the state representation contains reward vectors, this is trivially true for changes of reward location in an otherwise empty environment (Fig 3b3). However, it also holds for policies that require complicated trajectories avoiding multiple walls (Fig 3b4). These findings generalise across both discrete (Fig 3b) or continuous (Fig 3c) state and action representations. Together, these results emphasise that if hippocampal cells form a state-space for future learning, it is advantageous for them to build on existing relational structure (e.g. from cortical building blocks), as opposed to being learnt from scratch as is commonly assumed.

Compositional representations therefore allow for dramatic behavioural generalisation. In this view, the role of place cells is to bind together cortical representations into a memory, such that when the animal next visits this location they will be able to reactivate cortical representations that link to optimal actions. Paradoxically, because the agent does not need to have taken the action to build the memory, this is a memory of *future* behaviour. This memory of the future permits zero-shot inferences.

Latent learning and laying down memories of the future

Compositional reasoning therefore changes the computation required to produce good behaviour in a new environment. Instead of learning a new behavioural policy, we must lay down new memories, but critically we must lay down these memories *everywhere* in the environment. This is important because encountering a new wall should not only affect the policy at adjacent states. Instead, the presence of a wall will change the optimal policy everywhere. When the agent is next at a remote location, its state representation must reflect the fact that there is a wall between the agent and the reward.

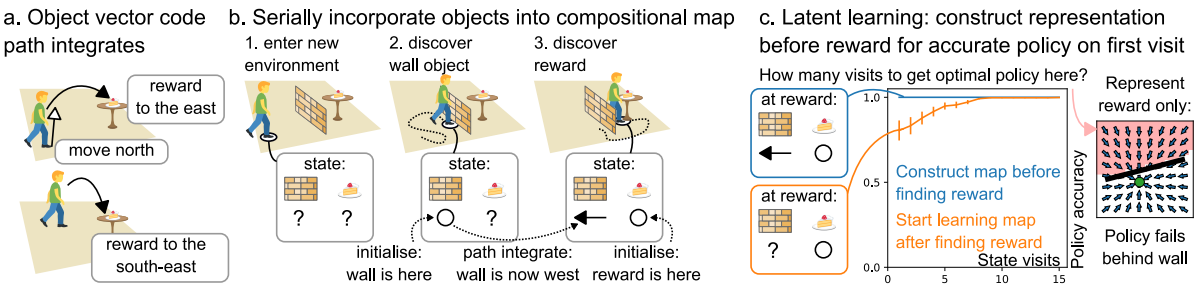

Fig 4. Latent learning through vector code path integration. **a**. The vector code (black arrowhead) path integrates: because it follows relational rules independent of the environment, the representation can be updated with respect to the agent's action (white arrowhead). For example, if a reward is to the east, and the agent goes north, the reward is now to the south-east. **b**. Path integration allows for serially incorporating objects (or walls or rewards) into the compositional map. The object-vector code is initialised on object discovery, and then carried along as the agent explores the environment. **c**. That means that the agent learns about the structure of the environment without being rewarded. Once it finds a reward, this latent learning allows access to optimal actions on the first visit to locations behind the wall (blue). Without latent learning, the agent needs to rediscover the wall to obtain the optimal policy behind the wall (orange).

Like credit assignment in RL, we therefore have to update state information at remote locations. But the content of these updates is structural information rather than reward expectations. Instead of value, we must transfer the newly discovered compositional features to each state in the environment. Critically, in compositional worlds the independent parts of the representation can be updated independently. In space, object-centric representations can be path integrated independently of allocentric representations (Fig 4a). If an agent is one step east of a reward and takes a step east, it is now two steps east of the reward. Each representation is determined by the last representation and the current action. Hence one approach is simply to keep track of the path integrated representation of each object (or wall or reward) it has encountered, as it traverses the environment. However, building these representations into memories alleviates the requirement to keep track of many variables at the same time. This is possible because the representation is compositional, which means that environmental features can be added one at a time.

This process of serially integrating information into a compositional state representation naturally accounts for results in latent learning. Here animals who have experienced the structure of the environment without rewards can rapidly develop optimal policies upon discovering rewards for the first time (Blodgett, 1929). Similarly, when our compositional agent explores the environment, it builds an increasingly complete state representation s, incorporating objects as they are encountered (Fig 4b). Then the agent finds a reward. In a simulation where the agent has already learned about the wall-vector representation, it only needs a single visit (to add the goal-vector to s) to any other state for access to the optimal policy (Fig 4c, blue). Without this latent learning, the agent needs to explore the whole environment again to accumulate the full state representation s (Fig 4c, orange).

In this simulation, we consider an agent that has explored an environment with a wall when discovering reward (Methods 5. Latent learning). To demonstrate the utility of latent learning, we compare our agent to an agent that only starts learning about the environment's objects after discovering the reward. The former has incorporated the wall in its state representation already and will have access to the optimal policy on the first visit to a new location, including those behind the wall. The latter needs to rediscover the wall, and will not know how to optimally avoid it until then (Fig 4c, right).

Replay builds memories efficiently

In the previous section we argued that path integration allows for serially incorporating objects into a compositional map during online exploration. However, this explanation of latent learning seems flawed. If the agent must path integrate representations by physically traversing the environment, and can only path integrate one or a few representations at a time, then it will need to traverse the environment many times to build a latent representation. To avoid this problem, rather than physically path integrating vector representations, the agent can *imagine* path integration in replay (Fig 5a). In this interpretation, replay can achieve credit assignment by exactly the same mechanism as it uses to build memories, because credit assignment is achieved by binding cortical representations into hippocampal memories.

This leads to clear untested predictions. Replay events should happen in the vector cells when an animal discovers a new environmental feature (such as an object or a reward). But critically, the replay events must bind object-centred representation to their correct locations in allocentric space. This means that allocentric representations must also path integrate in replay, simultaneously with the object-vector cells. A natural candidate for this allocentric representation is the grid cell representation, which is known to path integrate (Burak & Fiete, 2009; Sargolini et al., 2006). We therefore predict that replay events will involve simultaneous replay of grid cells and object-vector cells, where both cell populations replay to the same locations but in two different coordinate systems - global and object-centric respectively (Fig 5a). This also implies the resulting

hippocampal conjunctions (in this case a hippocampal object-vector-grid conjunction; landmark cells) at given locations can be active before ever physically visiting that location. Empirically, a landmark cell can be detected from its activity during online behaviour. We predict that some landmark cells will appear in replay, after discovering the corresponding object, before they appear in physical navigation.

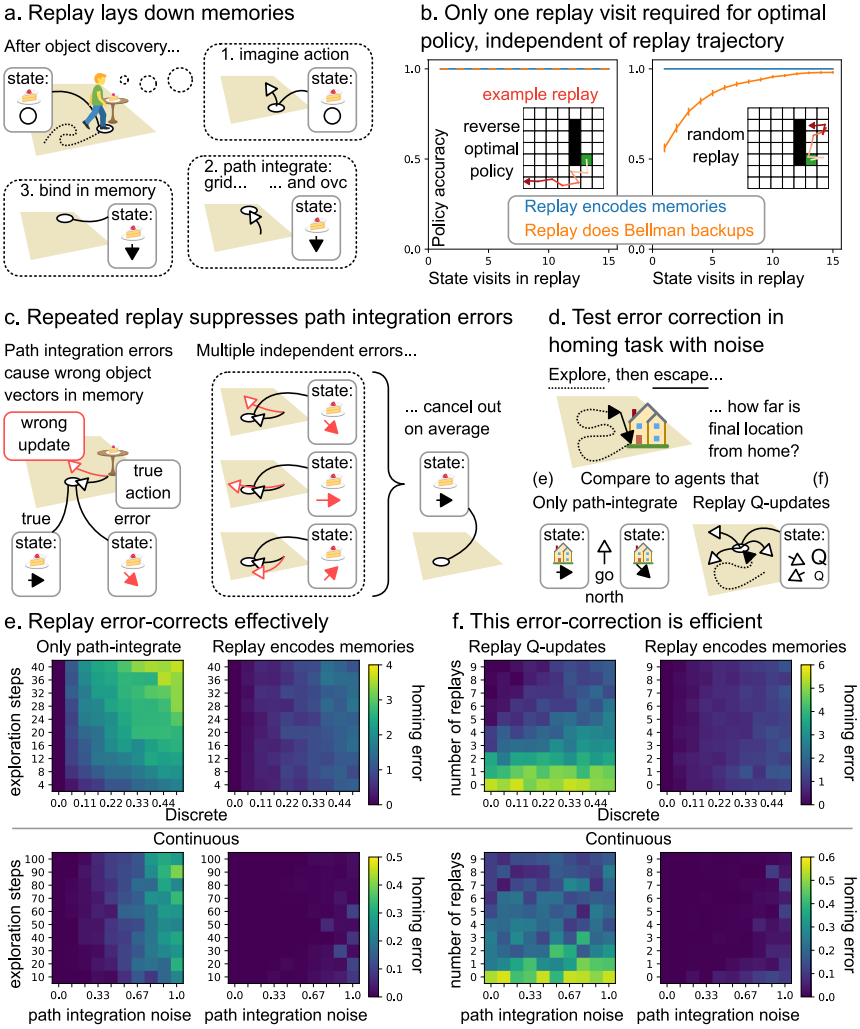

Fig 5. Replay builds compositional maps in memory. a. After discovering an object (or wall or reward), replay builds the compositional map in memory by path integrating the vector code and the grid code and binding them together to form new hippocampal memories at remote locations. **b**. Such replay (blue) provides optimal policies in a single replay visit. On the other hand, replay that performs Bellman backups for credit assignment (orange) requires a single visit for optimal actions *only* if the replayed trajectories follow the opposite optimal policy (left) but needs many more replay visits for random replay trajectories (right). **c.** Path integration is noisy. This can be mitigated by forming memories in multiple different replays since path integration errors are independent and average out. **d**. We test the error-correcting capacity of replay that encodes memories in a homing task with path integration noise. The agent starts from home, explores the environment, then needs to escape back home as quickly as possible. We compare the agent that encodes memories in replay to an agent that only path integrates a homing vector (panel e) and an agent that replays Q-updates to learn which actions are expected to lead towards home (panel f). **e**. For the agent that path integrates only (left), the homing error (distance from home after escape) increases for longer exploration and higher path integration noise. For the agent that encodes home-vector representation memories in replay, the error is greatly reduced. **f**. The agent that encodes home-vector memories in replay needs fewer replays to achieve lower homing errors than the agent that replays Q-updates.

Importantly, with noiseless path integration, the optimal new policy is constructed with a single visit to each state, whatever the trajectory of the replay (Methods 6. Constructive replay). This is unlike alternative interpretations of replay. For example, if replay performs Bellman back-ups (as in the dyna algorithm (Sutton, 1991)) it can only update the value of any state by comparing to its neighbour, so is exquisitely sensitive to the trajectory of the replay. To achieve convergence with a single step would require prescience (Fig 5b, left) as it would need to play out (in reverse) the new optimal policy that results from the updated values. Unlike Bellman back-ups, replaying for compositional memory requires only a single visit to each state, even if the trajectory is random (Fig 5b, right) (Gupta et al., 2010). This is because path integration through a world model (spatial or otherwise!) is trajectory-independent, and so the elements that get composed at each state are identical regardless of the replay's trajectory.

However, the reliance on path integration introduces a vulnerability: path integration is noisy, and errors accumulate over time. Replay will therefore build noisy, but unbiased, memories, where the noise can be reduced with repeated replays (Fig 5c). One attractive feature of this proposal is again that it aligns replay's role across credit assignment and memory. Here, multiple replays are required to consolidate the existing memory (Carr et al., 2011; Karlsson & Frank, 2009).

We simulate a noisy homing task where the agent explores an arena starting from a home location, until it encounters a threat and needs to return home as quickly as possible (Fig 5d; Methods 6. Constructive replay). Without replay, the homing error for the path integration agent increases both for higher noise and for longer exploration, as path integration errors accumulate (Fig 5e, left, top for discrete domains, bottom for continuous). With replay, homing errors of both causes are dramatically reduced (Fig 5e, right). Notably, Bellman backups are also susceptible to path integration errors as backups can be attributed to incorrect states. This can be ameliorated by sampling replay repeatedly (Fig 5f, left). Nevertheless, when replay is instead used to build compositional memories, smaller homing error is achieved with fewer replays (Fig 5f, right).

Optimal replay creates memories where they matter most

Requiring as few replays as possible for accurate behaviour is beneficial as replay takes time, and time is often needed for online behaviour (Agrawal et al., 2022; Jensen et al., 2023). Making the best use of each replay means prioritising certain replay trajectories over others. Intuitively, the highest priority replays are those that improve behaviour the most. This intuition has been formalised in the RL framework, where replay explicitly assigns credit to states through value backups, to successfully explain a wealth of empirically observed patterns of replay (Mattar & Daw, 2018). Constructive replay performs *implicit* credit assignment by binding object-centric representations, that come with pre-learned policies, to allocentric coordinates. Nevertheless, optimal constructive replay trajectories are qualitatively consistent with those predicted by the best value backups - albeit with a very different neural implementation.

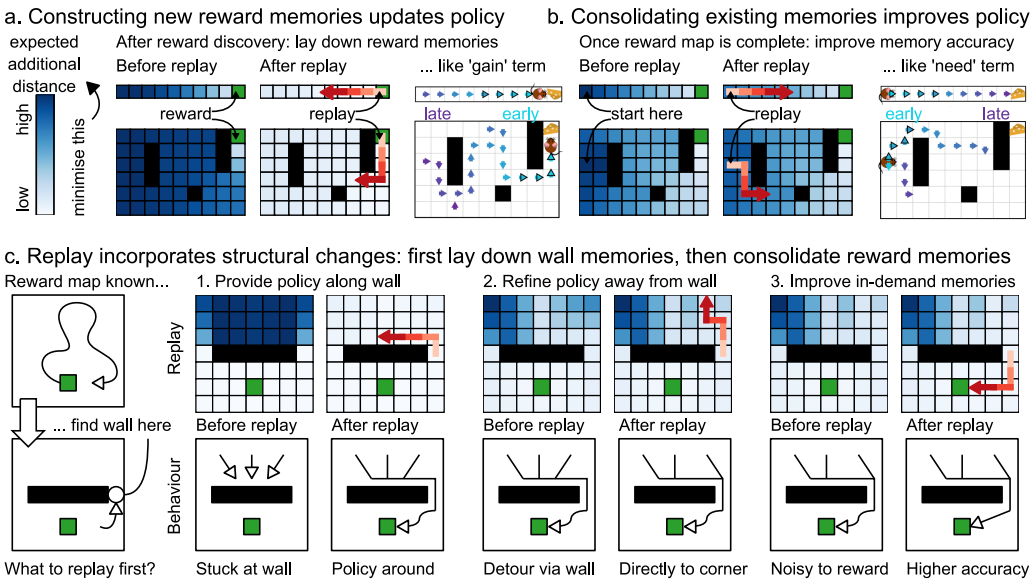

Fig 6. Optimal replay constructs, then consolidates maps. a. Optimal constructive replay minimises the total expected additional distance (the sum of additional distance per state plotted in blue). Upon finding a reward, that means laying down reward memories, producing trajectories similar to 'gain'-dominated value backups (Mattar & Daw, 2018). **b**. Once the reward map is sufficiently complete, optimal replays consolidate existing memories that will be needed often in the future, by encoding additional memories to error-correct path integration noise. Such replays play out along similar trajectories as 'need'-dominated value backups (Mattar & Daw, 2018). **c**. Constructive replay naturally accounts for structural changes, such as the discovery of a wall in the environment. Optimal replay encodes memories where they matter most. In this situation, that produces replays first along the wall to guide the policy around the wall, then away from the wall for a more direct policy, and then towards reward to improve the accuracy of memories that will be retrieved many times.

Here, we define optimal constructive replay as making *those* memories that are most impactful for shortening paths towards rewards from anywhere in the environment. Such replay forms new vector representation memories, or consolidates existing ones, at locations where these representations change the policy (for example by providing a reward vector) and are frequently retrieved (for example on common paths to reward). We find optimal replays by enumerating replay trajectories and evaluating how much the resulting memories decrease the expected distance towards the goal, averaged across the environment (Methods 7. Optimal Replay).

Optimal constructive replays play out along trajectories similar to backups that maximally increase future reward, because both aim to achieve the same goal: a map that supports selecting (ultimately) rewarding actions. Updates to the map take place in locations that have the greatest impact on the overall policy, regardless of the representation that underlies that policy. Upon reward discovery, such optimal updates distribute the reward information back through the environment (Fig 6a), whether through value backups or constructive replays. Once complete, the reward map can be further improved by consolidating the existing memories: additional constructive replays suppress path integration errors, along similar trajectories as 'need'-driven value backups (Fig 6b)(Mattar & Daw, 2018). These examples illustrate that the principle of replaying whatever improves the policy most can be applied irrespective of the replay mechanism: trajectories compatible with optimal value backups are often compatible with optimal state-space construction too.

However, that replay mechanism does sharply distinguish the two. Value backups treat the map, whether in hippocampus or cortex, as static, and change striatal synapses to reflect state values. Constructive replays change the hippocampal map itself. A new (replayed) composition that combines a particular vector and grid code recruits a new hippocampal representation - the conjunction between the two. That means that, in

addition to reward changes, constructive replay also naturally accounts for structural changes (Fig 6c). We simulate replays when an agent discovers a new wall in a familiar environment with a rewarded location. Optimal replay first runs along the wall (opposite side to the reward) while encoding wall-vector memories. Now all paths behind the wall are successfully rerouted around the wall, though they still go directly to the wall first, and then around. To make these trajectories less bendy, and more efficient, the next replay updates state-representations progressively further away from the wall, so that paths end up going directly to the edge of the wall. Finally, optimal replay consolidates the reward memories that are retrieved most often: those on the path from wall to reward. While no recordings of replay events exist in such situations, exactly these progressively less bendy behavioural trajectories are observed in behaviour when barriers are placed between a start and goal (home) location in a homing task (Shamash et al., 2021).

Composing non-spatial and hierarchical building blocks

Constructive replay thus carries structural information beyond just reward to compose representations that imply actions in remote locations. Importantly, this works equally well for non-spatial building blocks. That is important because empirically, hippocampus supports non-spatial reasoning (Aronov et al., 2017; Dusek & Eichenbaum, 1997), for example through representation of hierarchy (Kjelstrup et al., 2008; Shapiro et al., 1997) and context (McKenzie et al., 2014). Computationally, non-spatial composition opens up a whole range of new problems to which the same principles can be applied. These problems need to a) decompose into building blocks so a common function $f([z^1, z^2, z^3, \dots]) = a$ maps states to optimal actions, and b) allow independent forward models for each building block ${z^i}_{t+1} = g({z^i}_t, a_t)$ so building blocks can be replayed separately and then recombined in memory. Path integration is a specific spatial example of such forward models; more generally, they can be learned to capture a broad variety of environment dynamics.

We now show that our model works in a hierarchical task that mixes space and non-space (Methods 8. Non-spatial replay). This task consists of multiple spatial rooms (one of which contains a reward) joined together into a (non-spatial) loop by doors, but where the doors are at random spatial locations within each room and behave like teleports (i.e. the locations of the doors are unrelated to the rooms they transition to; Fig 7a). We provide a state representation that composes both spatial within-room vector codes, towards doors (d^1, d^2) or reward (r), and a non-spatial between-room vector code, that specifies the number of rooms between the current and the rewarding room (c). Importantly the d^1, d^2 and r representations are reused across different rooms. As before we train a neural network $f([d^1, d^2, r, c])$ to predict optimal actions on tasks with many different room sizes and door locations, and test on an entirely new hierarchical environment. We see that the network generalises to unseen configurations and provides accurate policies in each room towards the rewarded room (Fig 7b). This highlights generalisation across the hierarchy as anything learned in one room applies in another; alternative approaches like hierarchical RL would need to learn separate policies (or *options*) in each room.

In the situation described above, we provided the state representation. An agent, however, can only initialise a reward vector code (or rewarding room code) after actually observing a reward in one of the rooms. Similarly to before, we use replay to propagate this knowledge within the rewarded room (i.e. replay r vectors), but additionally replay across rooms (via c). This replay is thus decoupled over space (r) and non-space (c vectors). Since the replay of c is at a hierarchical level, a single replay suffices to obtain the optimal policy in all un-rewarded rooms (Fig 7c). The non-spatial between-room vector c can be viewed as a contextual signal that modulates behaviour depending on the state of the environment (like chopping or frying, depending on the current stage of a recipe). And because its forward model can implement arbitrary non-spatial dynamics (such as the non-spatial loop), it could guide actions in game environments that are currently challenging for AI agents (Wang et al., 2021).

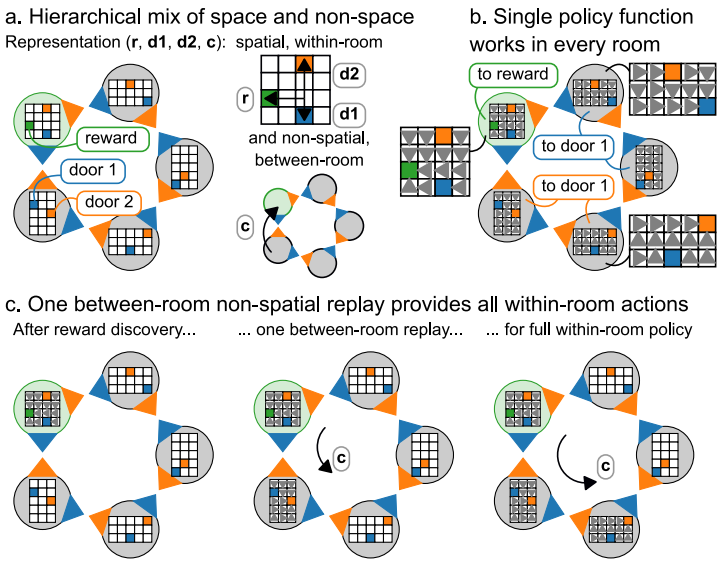

Fig 7. Policy and replay over hierarchical non-spatial building blocks. a. An example hierarchical environment that mixes space and non-space, consisting of rooms on a non-spatial loop. One room contains reward (green); each room contains two doors (blue and orange) at random locations that transition the agent to the next room along the non-spatial loop. We provide a full state representation that consists of within-room spatial vectors to reward and doors (r, d^1, d^2) and a between-room non-spatial vector code (c) to the rewarding room. **b**. A policy mapping $f([d^1, d^2, r, c]) = a$ learned across many hierarchical environments provides optimal actions in unseen new configurations, using the same within-room vector codes (d^1, d^2, r) in each room. **c**. Upon finding a reward, and replaying its within-room location, a single between-room replay propagates the rewarding room-vector code, c, thus providing the optimal policy to each of the other rooms.

DISCUSSION

Recent results have suggested a path towards building a formal understanding of neural responses in flexible behaviours. By assuming that the hippocampal formation builds a state-space (cognitive map) from sequential observations, these models have not only revealed computational insights into the hippocampal involvement in reinforcement learning (Piray & Daw, 2021; Stachenfeld et al., 2017), but also accounted for a variety of single neuron responses (George et al., 2021; Whittington et al., 2020). However, while this has the potential to bring formal explanations to an array of new scenarios it is not clear how these models relate to classic hippocampal functions such as episodic memory (Addis et al., 2007; Scoville & Milner, 1957), scene construction (Hassabis & Maguire, 2007), or imagination. Indeed, hippocampal patients are impaired not only in navigation, but also in scene recognition (Graham et al., 2010) and imagination of future and fictitious scenes (Hassabis et al., 2007; Mullally et al., 2012; Rosenbaum et al., 2009). In this work, we propose a framework where these disparate functions can be expressed in the same formal language. We formalise hippocampal state-spaces as compositions of reusable building blocks. We have shown that this affords flexibly generalising behaviour to new situations on first encounter, by simply rearranging previously learned components.

We propose that hippocampal cells provide conjunctions of pre-learnt building blocks, specifying their arrangement in the current experience. This allows us to reinterpret several hippocampal phenomena and offer new computational roles for others. For example, hippocampal place and landmark cells are conjunctions of grid cells with sensory cells (Whittington et al., 2020) and object-vector cells respectively. This means that hippocampus no longer needs to learn transitions itself, as these are inherited from the building block dynamics. Furthermore, we show that forming memories of these conjunctions (compositions) during explorations, provides the ideal state-space for *future* behaviour when observing the reward (latent learning).

Importantly, if the building blocks have a forward model then memories can be formed offline in replay. This enables an agent to efficiently build a compositional state-space for future behaviour. The idea that replay serves future behaviour has already proven influential (Mattar & Daw, 2018). Here we show how a different underlying mechanism can provide rapid zero-shot credit assignment, and hence generalisation. This constructive interpretation of replay naturally extends to changes in task structure as well as reward, and integrates these ideas with the compositional nature of hippocampal/entorhinal representations.

It is notable that in our model the state-space is simply the combination of appropriate *cortical* building blocks. The role of the hippocampus in the model is one of memory and replay. Access to this memory/replay system prevents the model from having to continually track all the elements of the forward model. Instead it can track one at a time (adding to existing state representations stored in memory) and can do so in replay rather than during behaviour (making online behaviour instantaneous). This minimal role for hippocampus is appropriate when worlds can be composed perfectly from existing primitive building blocks. By contrast, in models where hippocampus learns the transition structure (George et al., 2021; Stachenfeld et al., 2017), any transitions can be modelled (after learning). One intriguing possibility is that both systems are at play. Compositional inference gives rapid flexible approximate behaviour, which is nuanced by modelling of transitions. In such a system another possible role for replay is to build new cortical primitives when experiences have been poorly modelled by the existing repertoire.

While we have elucidated a role for replay in the online setting, this framework also suggests a neural interpretation of the two previously proposed roles for replay in sleep (Ellis et al., 2020): building cortical primitives (as above), and learning the policy on the compositional state-space. Here replay could help learn compositional policies by generating training examples for policy mapping f. By sampling random vector representations for walls and rewards, the agent dreams up arbitrary new environment configurations. Instead of encoding memories, it then replays to simulate trajectories in those imagined configurations. A successful rollout that reaches reward provides a training pair of the sampled vector code with an optimal action. This is a natural extension to the idea of learning an inverse model during sleep (Helmholtz machine (Dayan et al., 1995)), but it is particularly powerful for compositional forward models as “sleep training” can include samples that have never been experienced (Ellis et al., 2020).

Notably, we do not learn any building block representations in this paper, but instead assume representations in the form of a grid code for space and vector codes for the other building blocks. The ideas in this paper are not limited to grid and vector representations but apply to any forward models that can be decomposed into sub-models where actions have independent consequences. It will be particularly powerful to learn these forward models from experience - another possible role for sleep replay. Critically, our model assumes that these representations can be bound into hippocampal conjunctions such that any state in one elemental model can be bound with any state in another (the wall can be at any (x, y)-location). For this binding to work, it is likely that different elemental models should be expressed in different neural populations (disentanglement). Indeed, previous work has also shown that this is also the most energy efficient form of representation (Whittington, Dorrell, et al., 2022).

Our idea of constructing behaviour from reusable building blocks is related to meta-RL. In its broadest sense, meta-RL proposes a system that slowly learns over many episodes to train a fast within-episode learning system (learning to learn (Harlow, 1949)). Our compositional building blocks, and function f, permit fast within-episode learning, and we have assumed the building blocks are learned over many episodes. The implementation of meta-RL that has so far captured neuroscience, however, is where the dopamine system slowly trains prefrontal recurrent networks to solve sequential tasks (Wang et al., 2018). The key difference between our proposition is that we utilise conjunctive hippocampal representations (and Hebbian memories) for *explicit* building block compositions, whereas presumably a form of implicit composition is taking place in

the meta-RL prefrontal models. Lastly, we note that transformer neural networks (Vaswani et al., 2017), the state-of-the-art meta-learners (Adaptive Agent Team et al., 2023), are computationally closely related to hippocampal compositions (Whittington, Warren, et al., 2022).

Interestingly, while we have not explicitly tried to solve hierarchical RL, our framework nevertheless provides ingredients that may prove important. Hierarchy splits a big problem into smaller subproblems which may share structure and require similar solutions; this is exactly what our policy function f learns from building block codes, since they naturally generalise across subproblems. This is unlike the options framework (Botvinick, 2012) and related approaches like (Saxe et al., 2017), which only partially exploit hierarchical structure, as each smaller subproblem still needs to be solved individually. Additionally, because representations in our model are factorised between levels, new pieces of information about one hierarchical level immediately permeate to all other levels.

Extreme generalisation by composition is a fundamental property of human and animal cognition. Whilst this has been self-evident in cognitive science for many decades, it is typically ignored in computational neuroscience. In this work, we have taken this notion seriously and tried to align it with a series of properties of hippocampal function; some long-known - memory, construction and conjunctive coding; and some more recently discovered - state-spaces for controlling behaviour. As the community attempts to find formal descriptions for computations underlying increasingly rich and complex behaviours, we believe that compositional reasoning will play an increasingly important role in future models, not only of cognition but also of neural responses.

Acknowledgements
We thank Alon Baram and Kris Jensen for helpful comments on earlier drafts of the manuscript. We thank the following funding sources: Sir Henry Wellcome Post-doctoral Fellowship (222817/Z/21/Z) to J.C.R.W.; and Wellcome Principal Research Fellowship (219525/Z/19/Z), Wellcome Collaborator award (214314/Z/18/Z), and JS McDonnell Foundation award (JSMF220020372) to T.E.J.B.. The Wellcome Centre for Integrative Neuroimaging and Wellcome Centre for Human Neuroimaging are each supported by core funding from the Wellcome Trust (203139/Z/16/Z, 203147/Z/16/Z).

METHODS

A Python implementation of all models and simulations is available at https://github.com/jbakermans/state-space-composition. Here, we first describe the modelled worlds and agents in general, and then provide details of each figure's simulations.

1. Discrete and continuous worlds

We implement agents in continuous and discrete environments. Although similar in principle, these two types of agents and environments are slightly different in their practical implementation. The discrete agent behaves on a graph, whereas the continuous agent behaves in a two-dimensional Euclidean space. Both discrete and continuous settings are deterministic Markov Decision Processes (MDP).

The **discrete** environments are defined by a set of locations and actions as in a deterministic MDP. All results here assume rectangular square grid worlds, with actions 'North', 'East', 'South', 'West'. We generate environments by adding walls and rewards on top of these regular grids. We represent rewards with a population of object vector cells, where each cell fires at a specific vector relation to the reward, i.e. a cell that fires 1 step to the East and so on. We represent *each* wall with two populations of object vector cells - each vector population centred on one of the wall ends. In more details, we calculate the object vector population activity at location x relative to object o by concatenating the vectors of one-hot encoded distance from x to o along each action, with -1 distance for actions in the opposite direction. For example, for an object 1 step east and 3 steps south from x, the representation is $concatenate(onehot(-1), onehot(1), onehot(3), onehot(-1))$. Thus at any location, there are 4 vector cells active in the whole population - one for each action. In square grid worlds this representation has redundancy, because east is the opposite of west and north the opposite of south, but this setup allows for accommodating any type of non-grid graph too.

In **continuous** environments, locations become continuous (x, y)-coordinates, and actions are steps in a continuous direction. We place walls and rewards within a square 1m x 1m arena, at random locations and orientations. Again, we represent rewards by a single population of object vector cells and walls by two populations, one centred on each wall end. A single object vector cell is defined by a 2-dimensional gaussian firing field, tuned to a specific distance and direction from its reference object; the firing fields of the full object vector cell population are distributed on a square grid centred on the object. We thus calculate the object vector population activity at location x relative to object o by evaluating the population of gaussians centred on o at x.

2. Agent: tracking representations

Our agent does not have access to these vector representations when it enters a new environment (except in Fig 3 where we provide the full state representation). It needs to discover objects first. During exploration the agent observes its location and initialises a vector representation on object discovery, i.e. sets $concatenate(onehot(0), onehot(0), onehot(0), onehot(0))$ at that location x. From then on, it updates the vector representation on each transition. This update combines two components: 1) it path integrates its previous vector representation with respect to its action, and 2) it retrieves any existing vector representations previously stored in memory based upon the current observed location (the memory links vector representations to location representations). It then stores the updated representation at the new location in memory.

To implement this process we use two practical abstractions from a full neural system. First, we use a direct location signal (an id of the location), instead of a neural grid code from which location is traditionally thought to be decoded from. Second, we instantiate memory as a key-value dictionary, rather than a hippocampal attractor network that stores conjunctions (these are in fact directly relatable to each other (Ramsauer et al., 2021)). These abstractions do not change any model principles but keep the implementation simple.

The **discrete** agent initialises an object's vector representation when it is at a location adjacent to it. Then on each step, it path integrates the representation - but due to path integration noise, the *represented* vector relation might diverge from the *true* vector relation. We model this noise as a probability distribution p_{PI} over the updated representation, so that it reflects either the correct transition with probability $(1 - e_{PI})$ or one of the neighbours with probability e_{PI}. In addition to path integration, as stated above, the agent relies on memory to update its vector representation. After observing its new location, it retrieves vector representations inferred there previously. That produces another probability distribution p_M over represented vector relations, with the probability of a vector representation proportional to the number of retrieved memories of that representation. The agent then samples the final updated representation from the weighted sum of the path integrated and memory-retrieved distributions: $s \sim w \cdot p_{PI} + (1 - w) \cdot p_M$. Finally, it stores the sampled representation in memory at the new location.

The **continuous** agent tracks representations in a similar fashion. It discovers an object when it comes in range (5 cm), initialises the corresponding vector representation, and from then on updates it after every step. We model path integration errors in continuous space by adding gaussian noise to the direction and step size of the update to get a path integrated vector representation s_{PI}. Again, the agent combines this path integrated representation with a representation that is retrieved from memory. Continuous memories store vector representations at continuous locations; the agent retrieves all memories created within a cutoff distance (5 cm) from its new location, then weights the vector representations stored in these memories with a softmax over the cosine similarities between the memory location and the new location to obtain s_M. The updated representation is the weighted sum of path integration and memory-retrieval: $s = w \cdot s_{PI} + (1 - w) \cdot s_M$. The agent then encodes this inferred representation in memory at the new location.

3. Conjunctive memories

We propose that hippocampal representations are conjunctions of building block representations, and that these conjunctions can be stored as memories in hippocampal weights (Fig 2). While it is possible to implement a conjunctive representation in different ways, here we use an outer product representation, i.e. the conjunction c between representations a and b will have cells corresponding to the product of all pairs of a and b cells. So if a has 3 cells, and b has 4 cells, then c will have 12 cells. To demonstrate what conjunctive representations (e.g. between spatial representations and sensory representations) would look like in hippocampal recordings, we simulate populations of grid cells, object vector cells, and sensory neurons. Each of these neurons is defined by one or multiple spatial gaussian firing fields, arranged on a triangular grid (grid cells), at fixed distance and direction from an object (object vector cells), or at a random environment location (sensory neurons). See example conjunctive representations in Fig 2.

Conjunctive hippocampal cells remap in very different ways to random hippocampal cells. To reproduce the non-random remapping demonstrated in the Tolman-Eichenbaum Machine (Fig 2d), we generate 5 grid cells and 4 sensory neurons in two different environments. Across environments, the simulated grid cells shift together (as real grid cell correlation structure is preserved), while the simulated sensory neurons rearrange randomly since different spatial environments have different sensory particularities. We calculate all hippocampal cell ratemaps, i.e. the $5 \times 4 = 20$ hippocampal conjunctions between grid cells and sensory

neurons in both environments, and then for all $20 \times 5 = 100$ place-grid pairs we find the firing rate of the grid cell at the peak location of the place cell ratemap in each environment.

To show how hippocampal landmark cells could result from conjunctions of object vector cells and grid cells (Fig 2f), we generate object vector cell ratemaps and perform the outer product with a grid cell representation. This leads to landmark cell responses since a grid cell's peaks do not always align every object (or object vector cell), and so each conjunctive cell will not necessarily be active around every object.

4. Policies that generalise

If an agent has access to the full compositional vector representation, it should be able to behave optimally even if it has never seen the particular composition before (Fig 3). We demonstrate this by learning a policy mapping $f(s) = a$ that maps an input state representation to an output optimal action. In discrete environments, the output action is a vector of probabilities across the four discrete actions; in continuous environments, the output action is a two-dimensional vector that contains the sine and cosine of the optimal direction. For the compositional vector representation, s is the concatenation of object vector population activities described above (e.g. rewards, walls etc). As a control, we also learn a mapping for a representation of absolute location ('traditional'), i.e. a representation that does not know about walls or rewards.

We implement the function f as a feedforward neural network, with three hidden layers (dimensions 1000, 750, 500 in discrete environments; 3000, 2000, 1000 in continuous environments) with rectified linear activations. For the wall representation (which is two populations of vector cells per wall), we include an additional single network layer (common to all walls) that takes in the two populations and embeds them (same embedding dimension as input dimension) before feeding them in to the first hidden layer of of f - this is not necessary for learning, but does speed it up. We then sample [state representation, optimal action] pairs as training examples from environments with just a reward and environments with a reward and multiple walls, and train the network weights in a supervised manner through backpropagation. To evaluate a learned mapping, we sample locations and then simulate a rollout that follows the learned policy, and calculate the fraction of locations from where following that policy leads to reward (within 5cm).

To test whether policies learning from a *single* environment generalise (Fig 3b1,2; Fig 3c1,2), we sample training examples from one environment (discrete: 1000 samples; continuous: 2500 samples) and test on the same environment, for 25 environments independently. To test whether policies learned from *many* environment generalise (Fig 3b3,4; Fig 3c3,4), we train the network on 25 environments in parallel (batched input), sampling training examples in each environment (discrete: 200 samples; continuous: 500 samples) before sampling a new set of 25 environments (100 times). We then test the learned mapping for a new set of 25 environments not included in training.

5. Latent learning

Upon entering a new environment, we allow the agent to obtain and update compositional representations of objects/walls even in the absence of reward - this is latent learning (Fig 4). To show the utility of latent learning (Fig 4c), we simulate agents with and without latent learning in a discrete environment with a wall and reward. We consider the situation where both agents have already explored the environment and found the wall, and now they have just discovered the reward. The agent that does latent learning then has a full vector representation of wall and reward, while the agent without latent learning only knows about the reward vector (as it did not obtain or update the wall representations when it saw the wall earlier). We simulate both agents as they continue their exploration. The latent learning agent can use path integration to continue updating wall/reward vector representations, and can use these to calculate the optimal policy, wherever it

goes even if it has never been there before. The non-latent learning agent, on the other hand, needs to rediscover the wall to incorporate it in its state representation before it can behave optimally from all locations. To calculate the difference in their optimality, for each location behind the wall (where the full vector representation is required for appropriate behaviour), we calculate whether the agent would be able to successfully navigate to the reward based on its current representation on every encounter. We average policy success across these locations on first, second, et cetera until fifteenth, encounter and repeat the simulation 25 times in different environments.

6. Constructive replay

Replay offers a way of carrying vector representations to remote locations, without having to physically navigate there (Fig 5). During replay, the agent imagines actions, path integrates location (grid cell) and vector (object vector cell) representations, and binds them together by encoding a new memory of the resulting combination. We model these replayed transitions like the ones in physical navigation, with two important differences: 1) the agent cannot observe the transitioned location like in behaviour, so there can be path integration errors in the memory location (the 'key' in the dictionary) as well as in the representation (the 'value'), and 2) it only relies on path integration to update representations during replay, without the memory retrieval.

First, we compare an agent that encodes memories in replay like this to an agent that instead carries out credit assignment by temporal difference learning through Q-updates (Fig 5b). We apply 'backwards' Q-updates (temporal discounting factor $\gamma = 0.7$, learning rate $\alpha = 0.8$), in the opposite direction of replay (i.e. after a replay transition from a to b we calculate a backup from b to a), to make credit assignment more efficient. We sample replay trajectories that start from the reward location and either extend out along a random policy (Fig 5b, right), or a reverse-optimal policy (Fig 5b, left). We then calculate for each step in the replay trajectories whether the currently learned policy, either according to the encoded vector representation or the Q-values, provides an optimal path to reward from that step's location. We aggregate policy success by the first, second, et cetera until fifteenth, replay visit to each location and average across locations, and repeat the simulation 25 times in different environments.

Then, we investigate the consequences of path integration noise (Fig 5e,f). We simulate a homing task, where the agent starts from home and initialises its home-vector representation, then explores the arena, until it needs to escape to home as quickly as possible. During exploration, the agent builds home-vector memories by replaying from the home location 5 times every 4 steps, and retrieves these memories when it is updating its current home-vector representation (i.e. using memories to reduce path integration noise). During escape, it selects actions that lead home according to its current home-vector representation. In two different experiments, we compare the agent to two different controls. The first control agent only path integrates its homing vector, without encoding or retrieving any memories (and without any replay). In this experiment (Fig 5e) we vary path integration noise and the total number of steps during exploration. The second control agent carries out Q-updates to learn actions (discretised direction in the continuous environment) expected to lead home in on-policy replays, and uses the learned Q-values to find the way home during escape. Because the current location can be observed from the environment, the read-out of Q-values during escape does not suffer from path integration errors, but path integration noise does affect the replayed Q-updates: during replay, the location to update Q-values for needs to be path integrated, so credit can get assigned to the wrong place. In this experiment (Fig 5f), we vary the path integration noise and the number of replays that the agent engages in every 4 steps.

7. Optimal replay

Given the proposed constructive function of replay, what should its content be (Fig 6)? We define a replay as optimal if it maximally reduces the expected additional distance to reward (as compared to before replay), summed across all locations. We search over all possible replays to find the replay trajectory that best improves this metric. Conceptually, there are two ways replay can improve the metric: Either by adding vector representations of previously missing elements into the compositional map, or by consolidating already existing representations to reduce path integration noise. We now describe how we calculate the expected additional distance, and show how replay can improve it by these two ways.

To fully elucidate how the memories formed in constructive replay improve expected additional distance, we need to think about the possible representations at each location. In particular, each location either has access to the *full* state representation, composed of vector representations for all objects/walls/rewards, or a *partial* state representation, where an element of the composition is missing - but replay can provide the missing element. Partial representations can have different consequences for policy optimality. Sometimes the resulting policy is still successful (for example, when the missing element is irrelevant for the policy, like a wall to the north when there is reward to the south), but other times the agent will get stuck. Replay must then choose which locations to visit to make *partial* state representations *full* or to further improve representations that are currently noisy.

There are multiple possible ways to formalise expected additional distance. We formalise it by comparing the optimal path versus the path taken by an agent that operates in three regimes related to whether it has access to *full* or *partial* representations. The first regime is when it has a *full* representation - in which case it can path integrate straight to the goal. The second regime is when it has a *partial* representation - then it can path integrate but may get stuck in states. The third regime is random behaviour - it can switch to this behaviour after it gets stuck. This means the agent will always eventually find the reward. The agent can start using a *partial* regime then transition into the *full* regime on visiting a state with a memory of the *full* state representation, or it can transition from *partial* to random if the agent gets stuck. We formally model the above using three absorbing Markov chains on the discrete environment. The dynamics of these Markov chain are specified by three transition matrices, defined by policies as $T_{ss'} = p(s'|s) \ = \ \Sigma_a p(a|s) \cdot \ p(s'|s,a)$: the transitions T^{full} for the policy given the full state representation, T^{part} for the policy given a partial state representation, and T^{rand} for the random exploration policy. The reward is an absorbing state for each of the chains. For T^{part} there are two additional types of absorbing state: locations where the partial policy gets stuck, for example when there is an unexpected wall in front of the reward, and locations where memories of the full state representation have been encoded. Together, these three chains allow us to analyse a process where an agent that does not have access to the full state representation follows the partial policy until it arrives at reward, arrives at a memory, or gets stuck. If it arrives at reward, it is done (1). If it arrives at a full representation memory, it switches to the dynamics of the full policy T^{full} that takes it to reward (2). If it gets stuck, it starts random exploration following T^{rand} until it either arrives at reward, or hits a memory that provides it with the full representation, so it can follow T^{full} to reward (3). The total expected distance for a location to reward thus becomes the sum of the expected distances of all these scenarios, weighted by their probabilities:

$d_{start \to reward}$ =

(1) $p^{part}(start \to reward) \cdot d^{part}{}_{start \to reward}$

(2) $+\Sigma_{mem} p^{part}(start \to mem) \cdot (d^{part}{}_{start \to mem} + d^{full}{}_{mem \to reward})$

(3) $+\Sigma_{stuck} p^{part}(start \to stuck) \cdot (d^{part}{}_{start \to stuck} + p^{rand}{}_{stuck \to reward} \cdot d^{rand}{}_{stuck \to reward}$
$+ \Sigma_{mem} p^{rand}{}_{stuck \to mem} \cdot [d^{rand}{}_{stuck \to mem} + d^{full}{}_{mem \to reward}])$

To evaluate this expected distance we need to calculate the absorption probability and expected time until absorption for a given chain and a given absorbing state. To get these, standard Markov chain analysis separates the transition matrix T into the dynamics between transient (non-absorbing) states Q and transitions from transient to absorbing states R, and defines fundamental matrix $N = (I_t - Q)^{-1}$, where I_t is

the identity matrix with dimension number-of-transient-states. The probability of getting absorbed at state j, starting from state i, is given by the (i, j)-element of $B = NR$. The expected time until absorption *anywhere* starting from state i is given by the sum of the i-th row of N. To get the expected time until absorption *at a specific absorbing state* k, we calculate the sum across columns of the fundamental matrix M of an adjusted transition matrix U that expresses transition probabilities *given eventual absorption at* k: $U_{ij} = B_{ik}T_{ij}/B_{jk}$ (Clyde, 2022).
The above considers replay that provides missing elements to the compositional map in memory. But because memories are noisy, due to path integration errors, replay can also improve the policy by encoding memories for elements where these already exist. Repeated replay of existing memories improves the accuracy of those memories. To model this, we additionally inject noise into the transition matrices T^{full} and T^{part}. Because this noise is caused by path integration errors, we model it by mixing the policy at one location with the policy of the neighbours, plus a fixed amount of random policy: $\pi' = m\pi_{noise} + (1 - m)\pi$. Here π' is a vector of action probabilities, π is the noise-free policy vector, and π_{noise} is the mean of the policies across all neighbours and a random policy with equal probability for each action. The amount of mixing m, which determines the policy noise level, is lowered from 0.2 to 0.1 when replay creates an additional memory of an existing representation at that location (i.e. goes from 1 memory to 2 memories).

To actually calculate the optimal replay trajectories, we first calculate the expected disxtance to reward for each location, subtract the true distance to reward, then sum across all locations. Then we sample all possible 5-step replay sequences starting from the agent's current location and recalculate this total expected additional distance given the memories created in that replay. The optimal replay is the one that decreases the total expected additional distance the most.

8. Non-spatial replay

Finally, we show how our model of state-space composition can be extended to accommodate hierarchies of spatial and non-spatial structures (Fig 7). As an example of such an environment, we consider five discrete spatial regular square grid rooms, with shapes that are randomly sampled from [4x4, 3x5, 5x3], connected on a length-5 non-spatial loop. Now transitions exist on both hierarchical levels, within-room and between-room; in general, there can be many recursive levels where a location on the higher level corresponds to a whole environment on the lower level, and a lower-level action can take the agent to a different higher-level location. Here, the low-level rooms contain two doors, which when entered transition the agent to the adjacent high-level room, and one of the low-level rooms contains reward. Crucially the doors are located randomly within each room so their location tells you nothing about the high level action. We provide the agent a state representation that consists of within-room door and reward vectors d^1, d^2, r and between-room vector c that specifies the clockwise and anticlockwise distance towards the rewarding room.

Like before, we train a policy mapping $f(s) = a$, where the state representation input concatenates vector representations across hierarchical levels $s = [d^1, d^2, r, c]$ and the action output produces an 'primitive' action, on the lowest level of the hierarchy. Intuitively, it makes sense that this state representation affords correct policies: the c representation tells the agent which door to use, and the d^1, d^2 representations tells them how to get to that door - or directly to reward following r if already in the rewarding room. The agent can therefore reuse the same object vector cell population in every room (Fig 7b). We implement f as a feedforward neural network with one hidden layer of twice the size of the input dimension. To train f, we generate many different environments where the shapes of the rooms, the door locations, and the rewarding room and reward location are randomised. We train f through supervised learning by backpropagation on 500 samples of [state representation, optimal action] pairs from 25 of such environments in parallel, and repeat this 10 times with new environments.

Since door locations are different for each room, the agent utilises a low-level memory bank for each room that binds within-room vector representations to low-level locations. Additionally, the agent has a high-level memory bank that binds between-room vector representations to high-level rooms. This setup factorises the different levels of the hierarchy, and so replay can be independent for the different levels of the hierarchy.

When the agent has, through latent learning, built low-level memories of all door-vectors d^1, d^2 within every room before finding a reward, it has complete maps of the low-level environments but no optimal policy yet - it knows where the doors are, but not yet which door to take. But when it finds the reward in the rewarding room, it only needs to replay at the high-level to create a memory that binds the between-room reward vector c to the room to get access to the optimal policy (Fig 7c). Next time when it enters a room, it retrieves c from high-level memory and d^1, d^2 from the low-level memory bank for that room, which tells it both where the doors are and which door to approach.

9. Summary in double dactyl

Lego with state-spaces:
Cortical building blocks
bound through conjunction
in any new way.

Thus hippocampus is
combinatorially
building behaviour
like children's (re)play.

References

Adaptive Agent Team, Bauer, J., Baumli, K., Baveja, S., Behbahani, F., Bhoopchand, A., Bradley-Schmieg, N., Chang, M., Clay, N., Collister, A., Dasagi, V., Gonzalez, L., Gregor, K., Hughes, E., Kashem, S., Loks-Thompson, M., Openshaw, H., Parker-Holder, J., Pathak, S., … Zhang, L. (2023). *Human-Timescale Adaptation in an Open-Ended Task Space* (arXiv:2301.07608). arXiv. https://doi.org/10.48550/arXiv.2301.07608

Addis, D. R., Moscovitch, M., & McAndrews, M. P. (2007). Consequences of hippocampal damage across the autobiographical memory network in left temporal lobe epilepsy. *Brain*, *130*(9), 2327–2342. https://doi.org/10.1093/brain/awm166

Agrawal, M., Mattar, M. G., Cohen, J. D., & Daw, N. D. (2022). The temporal dynamics of opportunity costs: A normative account of cognitive fatigue and boredom. *Psychological Review*, *129*, 564–585. https://doi.org/10.1037/rev0000309

Anderson, M. I., & Jeffery, K. J. (2003). Heterogeneous Modulation of Place Cell Firing by Changes in Context. *Journal of Neuroscience*, *23*(26), 8827–8835. https://doi.org/10.1523/JNEUROSCI.23-26-08827.2003

Aronov, D., Nevers, R., & Tank, D. W. (2017). Mapping of a non-spatial dimension by the hippocampal-entorhinal circuit. *Nature*, *543*(7647), 719–722. https://doi.org/10.1038/nature21692

Blodgett, H. C. (1929). *The effect of the introduction of reward upon the maze performance of rats.* University of California.

Bostock, E., Muller, R. U., & Kubie, J. L. (1991). Experience-dependent modifications of hippocampal place cell firing. *Hippocampus*, *1*(2), 193–205. https://doi.org/10.1002/hipo.450010207

Botvinick, M. M. (2012). Hierarchical reinforcement learning and decision making. *Current Opinion in Neurobiology*, *22*(6), 956–962. https://doi.org/10.1016/j.conb.2012.05.008

Burak, Y., & Fiete, I. R. (2009). Accurate Path Integration in Continuous Attractor Network Models of Grid Cells. *PLOS Computational Biology*, *5*(2), e1000291.

https://doi.org/10.1371/journal.pcbi.1000291

Carr, M. F., Jadhav, S. P., & Frank, L. M. (2011). Hippocampal replay in the awake state: A potential substrate for memory consolidation and retrieval. *Nature Neuroscience*, *14*(2), Article 2. https://doi.org/10.1038/nn.2732

Clyde, D. (2022, December 17). *Answer to 'Expected time till absorption in specific state of a Markov chain'*. Mathematics Stack Exchange. https://math.stackexchange.com/a/4600370

Dayan, P., Hinton, G. E., Neal, R. M., & Zemel, R. S. (1995). The Helmholtz Machine. *Neural Computation*, *7*(5), 889–904. https://doi.org/10.1162/neco.1995.7.5.889

Deshmukh, S. S., & Knierim, J. J. (2013). Influence of local objects on hippocampal representations: Landmark vectors and memory. *Hippocampus*, *23*(4), 253–267. https://doi.org/10.1002/hipo.22101

Dusek, J. A., & Eichenbaum, H. (1997). The hippocampus and memory for orderly stimulus relations. *Proceedings of the National Academy of Sciences*, *94*(13), 7109–7114. https://doi.org/10.1073/pnas.94.13.7109

Eichenbaum, H., & Cohen, N. J. (2014). Can We Reconcile the Declarative Memory and Spatial Navigation Views on Hippocampal Function? *Neuron*, *83*(4), 764–770. https://doi.org/10.1016/j.neuron.2014.07.032

Ellis, K., Wong, C., Nye, M., Sable-Meyer, M., Cary, L., Morales, L., Hewitt, L., Solar-Lezama, A., & Tenenbaum, J. B. (2020). DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning. *ArXiv:2006.08381 [Cs]*. http://arxiv.org/abs/2006.08381

Etienne, A. S., & Jeffery, K. J. (2004). Path integration in mammals. *Hippocampus*, *14*(2), 180–192. https://doi.org/10.1002/hipo.10173

Foster, D. J., & Wilson, M. A. (2006). Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, *440*(7084), Article 7084. https://doi.org/10.1038/nature04587

Frank, L. M., Brown, E. N., & Wilson, M. (2000). Trajectory Encoding in the Hippocampus

and Entorhinal Cortex. *Neuron*, *27*(1), 169–178. https://doi.org/10.1016/S0896-6273(00)00018-0

Garvert, M. M., Dolan, R. J., & Behrens, T. E. (2017). A map of abstract relational knowledge in the human hippocampal–entorhinal cortex. *ELife*, *6*, 1–20. https://doi.org/10.7554/eLife.17086

Gauthier, J. L., & Tank, D. W. (2018). A Dedicated Population for Reward Coding in the Hippocampus. *Neuron*, *99*(1), 179-193.e7. https://doi.org/10.1016/j.neuron.2018.06.008

George, D., Rikhye, R. V., Gothoskar, N., Guntupalli, J. S., Dedieu, A., & Lázaro-Gredilla, M. (2021). Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps. *Nature Communications*, *12*(1), 2392. https://doi.org/10.1038/s41467-021-22559-5

Graham, K. S., Barense, M. D., & Lee, A. C. H. (2010). Going beyond LTM in the MTL: A synthesis of neuropsychological and neuroimaging findings on the role of the medial temporal lobe in memory and perception. *Neuropsychologia*, *48*(4), 831–853. https://doi.org/10.1016/j.neuropsychologia.2010.01.001

Gupta, A. S., van der Meer, M. A. A., Touretzky, D. S., & Redish, A. D. (2010). Hippocampal Replay Is Not a Simple Function of Experience. *Neuron*, *65*(5), 695–705. https://doi.org/10.1016/j.neuron.2010.01.034

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., & Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, *436*(7052), 801–806. https://doi.org/10.1038/nature03721

Harlow, H. F. (1949). The formation of learning sets. *Psychological Review*, *56*(1), 51–65. https://doi.org/10.1037/h0062474

Hassabis, D., Kumaran, D., Vann, S. D., & Maguire, E. A. (2007). Patients with hippocampal amnesia cannot imagine new experiences. *Proceedings of the National Academy of Sciences*, *104*(5), 1726–1731. https://doi.org/10.1073/pnas.0610561104

Hassabis, D., & Maguire, E. A. (2007). Deconstructing episodic memory with construction.

Trends in Cognitive Sciences, *11*(7), 299–306. https://doi.org/10.1016/j.tics.2007.05.001

Høydal, Ø. A., Skytøen, E. R., Andersson, S. O., Moser, M.-B., & Moser, E. I. (2019). Object-vector coding in the medial entorhinal cortex. *Nature*, *568*(7752), Article 7752. https://doi.org/10.1038/s41586-019-1077-7

Jensen, K. T., Hennequin, G., & Mattar, M. G. (2023). *A recurrent network model of planning explains hippocampal replay and human behavior* (p. 2023.01.16.523429). bioRxiv. https://doi.org/10.1101/2023.01.16.523429

Karlsson, M. P., & Frank, L. M. (2009). Awake replay of remote experiences in the hippocampus. *Nature Neuroscience*, *12*(7), Article 7. https://doi.org/10.1038/nn.2344

Kjelstrup, K. B., Solstad, T., Brun, V. H., Hafting, T., Leutgeb, S., Witter, M. P., Moser, E. I., & Moser, M.-B. (2008). Finite Scale of Spatial Representation in the Hippocampus. *Science*, *321*(5885), 140–143. https://doi.org/10.1126/science.1157086

Komorowski, R. W., Manns, J. R., & Eichenbaum, H. (2009). Robust Conjunctive Item–Place Coding by Hippocampal Neurons Parallels Learning What Happens Where. *Journal of Neuroscience*, *29*(31), 9918–9929. https://doi.org/10.1523/JNEUROSCI.1378-09.2009

Lever, C., Burton, S., Jeewajee, A., O'Keefe, J., & Burgess, N. (2009). Boundary Vector Cells in the Subiculum of the Hippocampal Formation. *Journal of Neuroscience*, *29*(31), 9771–9777. https://doi.org/10.1523/JNEUROSCI.1319-09.2009

Manns, J. R., & Eichenbaum, H. (2006). Evolution of declarative memory. *Hippocampus*, *16*(9), 795–808. https://doi.org/10.1002/hipo.20205

Mattar, M. G., & Daw, N. D. (2018). Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, *21*(11), Article 11. https://doi.org/10.1038/s41593-018-0232-z

McKenzie, S., Frank, A. J., Kinsky, N. R., Porter, B., Rivière, P. D., & Eichenbaum, H. (2014). Hippocampal representation of related and opposing memories develop within distinct, hierarchically organized neural schemas. *Neuron*, *83*(1), 202–215.

https://doi.org/10.1016/j.neuron.2014.05.019

Mittelstaedt, M.-L., & Mittelstaedt, H. (1980). Homing by path integration in a mammal. *Naturwissenschaften*, *67*(11), 566–567. https://doi.org/10.1007/BF00450672

Mullally, S. L., Hassabis, D., & Maguire, E. A. (2012). Scene Construction in Amnesia: An fMRI Study. *Journal of Neuroscience*, *32*(16), 5646–5653. https://doi.org/10.1523/JNEUROSCI.5522-11.2012

Muller, R. U., & Kubie, J. L. (1987). The effects of changes in the environment on the spatial firing of hippocampal complex-spike cells. *Journal of Neuroscience*, *7*(7), 1951–1968. https://doi.org/10.1523/JNEUROSCI.07-07-01951.1987

Nieh, E. H., Schottdorf, M., Freeman, N. W., Low, R. J., Lewallen, S., Koay, S. A., Pinto, L., Gauthier, J. L., Brody, C. D., & Tank, D. W. (2021). Geometry of abstract learned knowledge in the hippocampus. *Nature*. https://doi.org/10.1038/s41586-021-03652-7

O'Keefe, J. (1976). Place units in the hippocampus of the freely moving rat. *Experimental Neurology*, *51*(1), 78–109. https://doi.org/10.1016/0014-4886(76)90055-8

Packard, M. G., & McGaugh, J. L. (1996). Inactivation of Hippocampus or Caudate Nucleus with Lidocaine Differentially Affects Expression of Place and Response Learning. *Neurobiology of Learning and Memory*, *65*(1), 65–72. https://doi.org/10.1006/nlme.1996.0007

Piray, P., & Daw, N. D. (2021). Linear reinforcement learning in planning, grid fields, and cognitive control. *Nature Communications*, *12*(1), 4942. https://doi.org/10.1038/s41467-021-25123-3

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., Greiff, V., Kreil, D., Kopp, M., Klambauer, G., Brandstetter, J., & Hochreiter, S. (2021). *Hopfield Networks is All You Need* (arXiv:2008.02217). arXiv. https://doi.org/10.48550/arXiv.2008.02217

Rosenbaum, R. S., Gilboa, A., Levine, B., Winocur, G., & Moscovitch, M. (2009). Amnesia as an impairment of detail generation and binding: Evidence from personal, fictional, and semantic narratives in K.C. *Neuropsychologia*, *47*(11), 2181–2187.

https://doi.org/10.1016/j.neuropsychologia.2008.11.028

Russek, E. M., Momennejad, I., Botvinick, M. M., Gershman, S. J., & Daw, N. D. (2017). Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLOS Computational Biology*, *13*(9), e1005768. https://doi.org/10.1371/journal.pcbi.1005768

Sargolini, F., Fyhn, M., Hafting, T., McNaughton, B. L., Witter, M. P., Moser, M.-B., & Moser, E. I. (2006). Conjunctive Representation of Position, Direction, and Velocity in Entorhinal Cortex. *Science*, *312*(5774), 758–762. https://doi.org/10.1126/science.1125572

Saxe, A. M., Earle, A. C., & Rosman, B. (2017). Hierarchy Through Composition with Multitask {LMDP}s. *Proceedings of the 34th International Conference on Machine Learning*, *70*, 3017–3026.

Schapiro, A. C., Turk-Browne, N. B., Norman, K. A., & Botvinick, M. M. (2016). Statistical learning of temporal community structure in the hippocampus. *Hippocampus*, *26*(1), 3–8. https://doi.org/10.1002/hipo.22523

Scoville, W. B., & Milner, B. (1957). Loss of Recent Memory After Bilateral Hippocampal Lesions. *Journal of Neurology, Neurosurgery & Psychiatry*, *20*(1), 11–21. https://doi.org/10.1136/jnnp.20.1.11

Shamash, P., Olesen, S. F., Iordanidou, P., Campagner, D., Banerjee, N., & Branco, T. (2021). Mice learn multi-step routes by memorizing subgoal locations. *Nature Neuroscience*, *24*(9), Article 9. https://doi.org/10.1038/s41593-021-00884-8

Shapiro, M. L., Tanila, H., & Eichenbaum, H. (1997). Cues that hippocampal place cells encode: Dynamic and hierarchical representation of local and distal stimuli. *Hippocampus*, *7*(6), 624–642. https://doi.org/10.1002/(SICI)1098-1063(1997)7:6<624::AID-HIPO5>3.0.CO;2-E

Solstad, T., Boccara, C. N., Kropff, E., Moser, M.-B., & Moser, E. I. (2008). Representation of Geometric Borders in the Entorhinal Cortex. *Science*, *322*(5909), 1865–1868. https://doi.org/10.1126/science.1166466

Stachenfeld, K. L., Botvinick, M. M., & Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature Neuroscience*, *20*(11), 1643–1653. https://doi.org/10.1038/nn.4650

Sun, C., Yang, W., Martin, J., & Tonegawa, S. (2020). Hippocampal neurons represent events as transferable units of experience. *Nature Neuroscience*, *23*(May). https://doi.org/10.1038/s41593-020-0614-x

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, *2*(4), 160–163. https://doi.org/10.1145/122344.122377

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning, second edition: An Introduction*. MIT Press.

Tulving, E. (1985). How many memory systems are there? *American Psychologist*, *40*, 385–398. https://doi.org/10.1037/0003-066X.40.4.385

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. *Nips*. https://doi.org/10.1017/S0140525X16001837

Wang, J. X., King, M., Porcel, N., Kurth-Nelson, Z., Zhu, T., Deck, C., Choy, P., Cassin, M., Reynolds, M., Song, F., Buttimore, G., Reichert, D. P., Rabinowitz, N., Matthey, L., Hassabis, D., Lerchner, A., & Botvinick, M. (2021). *Alchemy: A benchmark and analysis toolkit for meta-reinforcement learning agents* (arXiv:2102.02926). arXiv. https://doi.org/10.48550/arXiv.2102.02926

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., Hassabis, D., & Botvinick, M. (2018). Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience*, *21*(6), 860–868. https://doi.org/10.1038/s41593-018-0147-8

Whittington, J. C. R., Dorrell, W., Ganguli, S., & Behrens, T. E. J. (2022). *Disentangling with Biological Constraints: A Theory of Functional Cell Types* (arXiv:2210.01768). arXiv. https://doi.org/10.48550/arXiv.2210.01768

Whittington, J. C. R., Muller, T. H., Mark, S., Chen, G., Barry, C., Burgess, N., & Behrens, T.

E. J. (2020). The Tolman-Eichenbaum Machine: Unifying Space and Relational Memory through Generalization in the Hippocampal Formation. *Cell*, *183*(5), 1249-1263.e23. https://doi.org/10.1016/j.cell.2020.10.024

Whittington, J. C. R., Warren, J., & Behrens, T. E. J. (2022). *Relating transformers to models and neural representations of the hippocampal formation* (arXiv:2112.04035). arXiv. https://doi.org/10.48550/arXiv.2112.04035

Wood, E. R., Dudchenko, P. A., Robitsek, R. J., & Eichenbaum, H. (2000). Hippocampal Neurons Encode Information about Different Types of Memory Episodes Occurring in the Same Location. *Neuron*, *27*(3), 623–633. https://doi.org/10.1016/S0896-6273(00)00071-4