LETTER

https://doi.org/10.1038/s41586-018-0102-6

Vector-based navigation using grid-like representations in artificial agents

Andrea Banino[1,2,3,5]*, Caswell Barry[2,5]*, Benigno Uria[1], Charles Blundell[1], Timothy Lillicrap[1], Piotr Mirowski[1], Alexander Pritzel[1], Martin J. Chadwick[1], Thomas Degris[1], Joseph Modayil[1], Greg Wayne[1], Hubert Soyer[1], Fabio Viola[1], Brian Zhang[1], Ross Goroshin[1], Neil Rabinowitz[1], Razvan Pascanu[1], Charlie Beattie[1], Stig Petersen[1], Amir Sadik[1], Stephen Gaffney[1], Helen King[1], Koray Kavukcuoglu[1], Demis Hassabis[1,4], Raia Hadsell[1] & Dharshan Kumaran[1,3]*

Deep neural networks have achieved impressive successes in fields ranging from object recognition to complex games such as Go[1,2]. Navigation, however, remains a substantial challenge for artificial agents, with deep neural networks trained by reinforcement learning[3–5] failing to rival the proficiency of mammalian spatial behaviour, which is underpinned by grid cells in the entorhinal cortex[6]. Grid cells are thought to provide a multi-scale periodic representation that functions as a metric for coding space[7,8] and is critical for integrating self-motion (path integration)[6,7,9] and planning direct trajectories to goals (vector-based navigation)[7,10,11]. Here we set out to leverage the computational functions of grid cells to develop a deep reinforcement learning agent with mammal-like navigational abilities. We first trained a recurrent network to perform path integration, leading to the emergence of representations resembling grid cells, as well as other entorhinal cell types[12]. We then showed that this representation provided an effective basis for an agent to locate goals in challenging, unfamiliar, and changeable environments—optimizing the primary objective of navigation through deep reinforcement learning. The performance of agents endowed with grid-like representations surpassed that of an expert human and comparison agents, with the metric quantities necessary for vector-based navigation derived from grid-like units within the network. Furthermore, grid-like representations enabled agents to conduct shortcut behaviours reminiscent of those performed by mammals. Our findings show that emergent grid-like representations furnish agents with a Euclidean spatial metric and associated vector operations, providing a foundation for proficient navigation. As such, our results support neuroscientific theories that see grid cells as critical for vector-based navigation[7,10,11], demonstrating that the latter can be combined with path-based strategies to support navigation in challenging environments.

The ability to self-localize in the environment and to update one's position on the basis of self-motion are core components of navigation[13]. We trained a deep neural network to path integrate within a square arena (2.2 m × 2.2 m), using simulated trajectories modelled on those of foraging rodents (see Methods). The network was required to update its estimate of location and head direction using translational and angular velocity signals, mirroring those available to the mammalian brain[12,14,15] (see Methods; Fig. 1a, b). Velocity was provided as input to a recurrent network with a long short-term memory (LSTM) architecture, which was trained using backpropagation through time (see Methods and Supplementary Discussion), allowing the network to dynamically combine current input signals with activity patterns reflecting past events (see Methods; Fig. 1a). The LSTM projected to place and head direction units via a linear layer—units with activity defined as a simple linear function of their input (see Extended Data Fig. 1 for architecture). Importantly, the linear layer was subject to regularization, in particular dropout[16], such that 50% of the units were randomly silenced at each time step. The vector of activities in the place and head direction units, corresponding to the current position, was provided as a supervised training signal at each time step (see Methods; Extended Data Fig. 1). This form of supervision follows evidence that in mammals, place and head direction representations exist in close anatomical proximity to entorhinal grid cells[12] and emerge in rodent pups before the appearance of mature grid cells[17,18]. Equally, in adult rodents, entorhinal grid cells are known to project to the hippocampus and appear to contribute to the neural activity of place cells[19].

As expected, the network was able to path integrate accurately in this setting involving foraging behaviour (mean error after 15 s trajectory was 16 cm versus 91 cm for an untrained network, effect size 2.83, 95% confidence interval (CI) 2.80–2.86; Fig. 1b, c). Strikingly, individual units within the linear layer of the network developed stable spatial activity profiles similar to those of neurons within the entorhinal network[6,12] (Fig. 1d, Extended Data Fig. 2). Specifically, 129 of the 512 linear layer units (25.2%) resembled grid cells, exhibiting significant hexagonal periodicity (gridness[20]) versus a null distribution generated by a conservative field shuffling procedure (see Methods). The scale of the grid-patterns, measured from the spatial autocorrelograms of the activity maps[20], varied between units (range 28 cm to 115 cm, mean 66 cm) and followed a multi-modal distribution, consistent with empirical results from rodent grid cells[21,22] (Fig. 1e). To assess these clusters, we fit mixtures of Gaussians, finding the most parsimonious number by minimizing the Bayesian information criterion (BIC). The distribution was best fit by three Gaussians (means 47 cm, 70 cm, and 106 cm), indicating the presence of scale clusters with a ratio between neighbouring clusters of approximately 1.5, closely matching theoretical predictions[23] and also lying within the range reported for rodents[21,22] (Fig. 1e, Extended Data Fig. 3). The linear layer also exhibited units resembling head direction cells (10.2%), border cells (8.7%), and a small number of place cells[12] as well as conjunctions of these representations[20,24] (Fig. 1d, f, g, Extended Data Fig. 2).

To ascertain how robust these representations were, we retrained the network 100 times, in each instance finding similar proportions of grid-like units (mean 23%, s.d. 2.8%, units with significant gridness scores) and other spatially modulated units (Extended Data Fig. 3). Conversely, grid-like representations did not emerge in networks without regularization (for example, dropout, see Methods; also see ref. 25, Extended Data Fig. 4). Therefore, the use of regularization, including dropout (which has been viewed to be a parallel of noise in neural systems[16]), was critical to the emergence of entorhinal-like representations. Notably, therefore, our results show that grid-like representations reminiscent of those found in the mammalian entorhinal cortex emerge in a generic network trained to path integrate, contrasting with previous approaches using pre-configured grid cells[26] (see Supplementary Discussion).

[1]DeepMind, London, UK. [2]Department of Cell and Developmental Biology, University College London, London, UK. [3]Centre for Computation, Mathematics and Physics in the Life Sciences and Experimental Biology (CoMPLEX), University College London, London, UK. [4]Gatsby Computational Neuroscience Unit, University College London, London, UK. [5]These authors contributed equally: Andrea Banino, Caswell Barry. *e-mail: abanino@google.com; caswell.barry@ucl.ac.uk; dkumaran@google.com

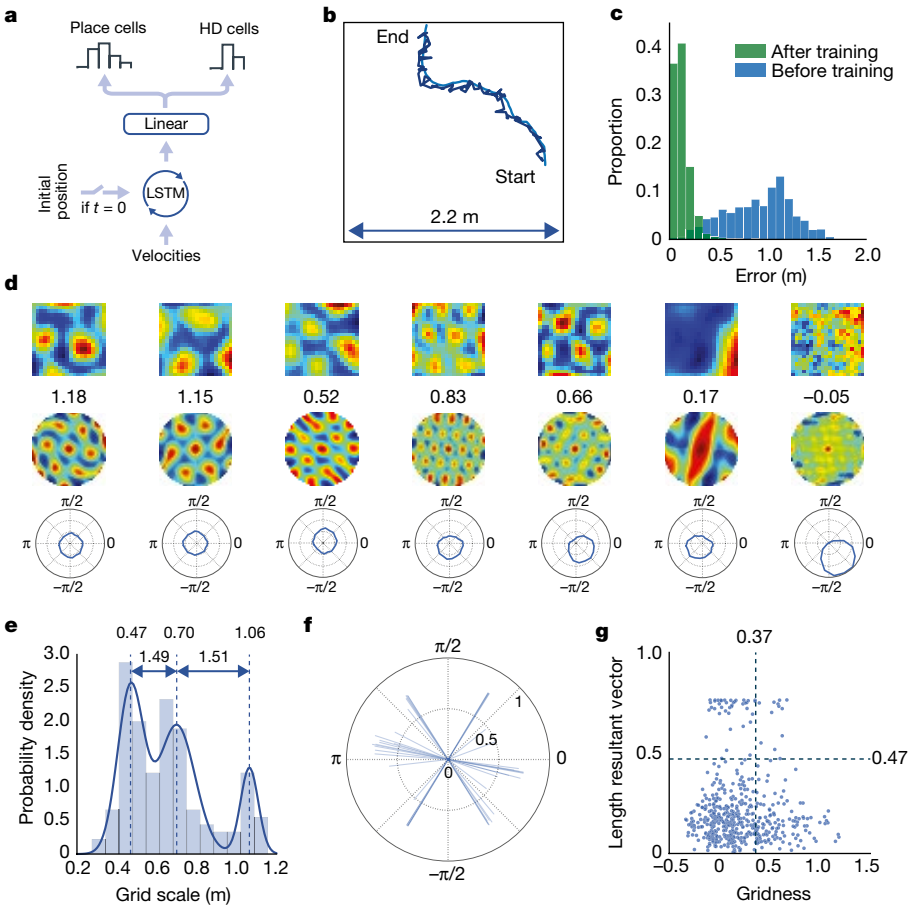


Fig. 1 | Entorhinal-like representations emerge in a network trained to path integrate. a, Schematic of network architecture (see Extended Data Fig. 1). **b**, Example trajectory (15 s); self-location decoded from place cells (dark blue) resembles actual path (light blue). **c**, Accuracy of decoded location before (blue) and after (green) training. **d**, Linear layer units exhibit spatially tuned responses resembling grid, border, and head direction cells. Top, ratemap shows activity over location; middle, spatial autocorrelogram of ratemap with gridness indicated; bottom, polar plot show activity versus head direction. **e**, Spatial scale of grid-like units ($n = 129$) is clustered. Distribution is more discrete than by chance (effect size, 2.98; 95% CI, 0.97–4.91) and best fit by a mixture of three Gaussians (centres 0.47, 0.70 and 1.06 m, ratios are 1.49 and 1.51). **f**, Directional tuning of the most strongly directional units ($n = 52$). Lines indicate length and orientation of resultant vector (see Methods), exhibiting six-fold clustering reminiscent of conjunctive grid cells. **g**, Distribution of gridness and directional tuning. Dashed lines indicate 95% confidence interval from null distributions (based on 500 data permutations); 14 (11%) grids exhibit directional modulation (see Methods). Similar results were seen in a circular environment (Extended Data Fig. 3).

Furthermore, our results are consistent with the view that grid cells represent an efficient and robust basis for a location code updated by self-motion cues[6–9].

Next, we sought to test the hypothesis that the emergent representations provide an effective basis function for goal-directed navigation in novel challenging and changeable environments, when trained through deep reinforcement learning. Entorhinal grid cells have been proposed to provide a Euclidean spatial metric and thus to support the calculation of goal-directed vectors, enabling animals to follow direct routes to a remembered goal, a process known as vector-based navigation[7,10,11]. Theoretically, the advantage of decomposing spatial location into a multi-scale periodic code, as provided by grid cells, is that the relative positions of two points can be retrieved by examining the difference in the code at the level of each scale—combining the modulus remainders to return the true vector[7,11] (Fig. 2a). However, despite the obvious utility of such a framework, experimental evidence for the direct involvement of grid representations in goal-directed navigation is still lacking.

To develop an agent with the potential for vector-based navigation, we incorporated the 'grid network' described above into a larger architecture that was trained using deep reinforcement learning (Fig. 2d, Extended Data Fig. 5). As before, the grid network was trained using supervised learning but, to better approximate the information available to navigating mammals, it now received velocity signals perturbed with random noise as well as visual input. Experimental evidence suggests that place cell input to grid cells corrects for drift and anchors grids to environmental cues[21]. To parallel this, visual input was processed by a 'vision module' consisting of a convolutional network that produced place and head direction cell activity patterns that were provided as input to the grid network 5% of the time—akin to a moving animal making occasional, imperfect observations of salient environmental cues[27] (see Methods; Fig. 2b, c and Extended Data Fig. 5). The output of the linear layer of the grid network, corresponding to the agent's current location, was provided as input to the 'policy LSTM', a second recurrent network that both controls the agent's actions and outputs a value function. Additionally, whenever the agent reached the goal, the 'goal grid code'—activity in the linear layer—was subsequently provided to the policy LSTM during navigation as an additional input.

We first examined the navigational capacities of the agent in a simple setting inspired by the classic Morris water maze (Fig. 2b, c; 2.5 m × 2.5 m square arena; see Methods and Supplementary Results). Notably, the agent was still able to self-localize accurately in this more challenging setting, where ground truth information about location was not provided and velocity inputs were noisy (mean error after 15-s trajectory, 12 cm versus 88 cm for an untrained network; effect size, 2.82; 95% CI, 2.79–2.84; Fig. 2e). Furthermore, the agent exhibited proficient goal-finding abilities, typically taking direct routes to the goal from arbitrary starting locations (Fig. 2h). Performance exceeded that of a control place cell agent (Fig. 2f, see Supplementary Results and

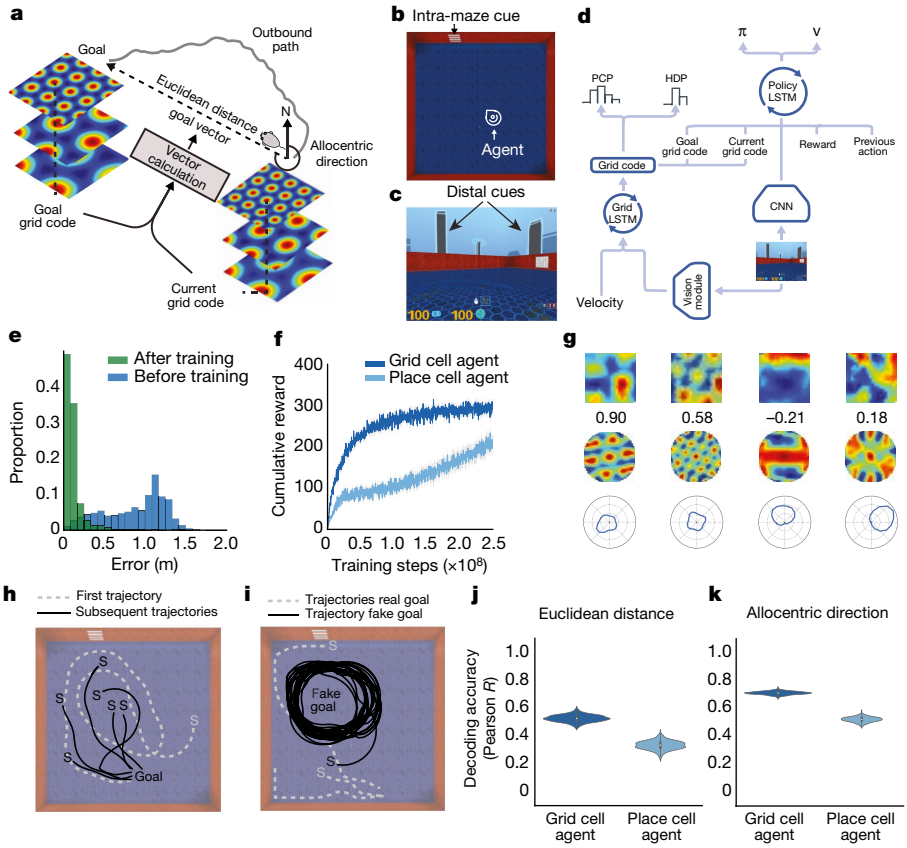


Fig. 2 | One-shot open field navigation to a hidden goal. a, Schematic of vector-based navigation. **b**, Overhead view of typical environment (icon indicates agent and facing direction). **c**, Agent view of **b**. **d**, Schematic of deep reinforcement learning architecture (Extended Data Fig. 5). PCP, place cell predictions; HDP, head direction cell predictions. **e**, Accuracy of self-location decoded from place cell units. **f**, Performance of grid cell agent and place cell agent (*y* axis shows reward obtained within a single episode, 10 points per goal arrival, grey band displays the 68% confidence interval based on 5,000 bootstrapped samples). **g**, As in Fig. 1, the linear layer develops spatial representations similar to entorhinal cortex. Left to right, two grid cells, one border cell, and one head direction cell. **h**, On the first trial of an episode, the agent explores to find the goal and subsequently navigates directly to it. 'S' denotes the starting location. **i**, After successful navigation, the policy LSTM was supplied with a 'fake' goal grid-code, directing the agent to this location where no goal was present. **j**, **k**, Decoding of goal-directed metric codes (that is, Euclidean distance and direction) from the policy LSTM of grid cell and place cell agents. The bootstrapped distribution (1,000 samples) of correlation coefficients are each displayed with a violin plot overlaid on a Tukey boxplot.

Methods), chosen because place cells provide a robust representation of self-location but are not thought to provide a substrate for long-range vector calculations[11]. We examined the units in the linear layer, again finding a heterogeneous population resembling those found in entorhinal cortex, including grid-like units (21.4%) as well as other spatial representations (Fig. 2g, Extended Data Fig. 6)—paralleling the dependence of mammalian grid cells on self-motion information[15,28] and spatial cues[6,21].

We next turn to our central claim, that grid cells endow agents with the ability to perform vector-based navigation, enabling downstream regions to calculate goal-directed vectors by comparing current activity with that of a remembered goal[7,10,11]. In the agent, we expect these calculations to be performed by the policy LSTM, which receives the current activity pattern over the linear layer (termed 'current grid code; Fig. 2d and Extended Data Fig. 5) as well as that present the last time the agent reached the goal (termed 'goal grid code') and uses them to control movement. Hence we performed several manipulations, which yielded four lines of evidence in support of the vector-based navigation hypothesis (see Supplementary Results).

First, to demonstrate that the goal grid code provided sufficient information to enable the agent to navigate to an arbitrary location, we substituted it with a 'fake' goal grid code sampled randomly from a location in the environment (see Methods). The agent followed a direct path to the newly specified location, circling the absent goal (Fig. 2i)—similar to rodents in probe trials of the Morris water maze (escape platform removed). Second, we demonstrated that withholding the goal grid code from the policy LSTM of the grid cell agent had a strikingly deleterious effect on performance (Extended Data Fig. 6c). Third, we demonstrated that the policy LSTM of the grid cell agent contained representations of key components of vector-based navigation (Fig. 2j, k), and that both Euclidean distance (difference in $r = 0.17$; 95% CI, 0.11–0.24) and allocentric goal direction (difference in $r = 0.22$; 95% CI, 0.18–0.26) were represented more strongly than in the place cell agent. Notably, a neural representation of goal distance has recently been reported in the mammalian hippocampus[29]. Finally, we provide evidence consistent with a prediction of the vector-based navigation hypothesis, namely that a targeted lesion (that is, silencing) to the most grid-like units within the goal grid code should have a greater adverse effect on performance and the representation of vector-based metrics (for example, Euclidean distance) than a sham lesion (that is, silencing of non-grid units; see Supplementary Results).

Having demonstrated the effectiveness of grid-like representations in optimizing one-shot goal learning in a simple square arena, we assessed the agent's performance in two challenging, procedurally generated multi-room environments, referred to as 'goal-driven' and 'goal-doors' (see Methods). Notably, these environments are challenging for deep reinforcement learning agents with external memory (Extended Data Fig. 7e, f, h, i and Supplementary Results). Again, the grid cell agent exhibited high levels of performance, was strikingly robust across a range of network hyperparameters (Extended Data Fig. 7a–c), and reached the goal more frequently than either control agents or a human expert—a typical benchmark for the performance of deep

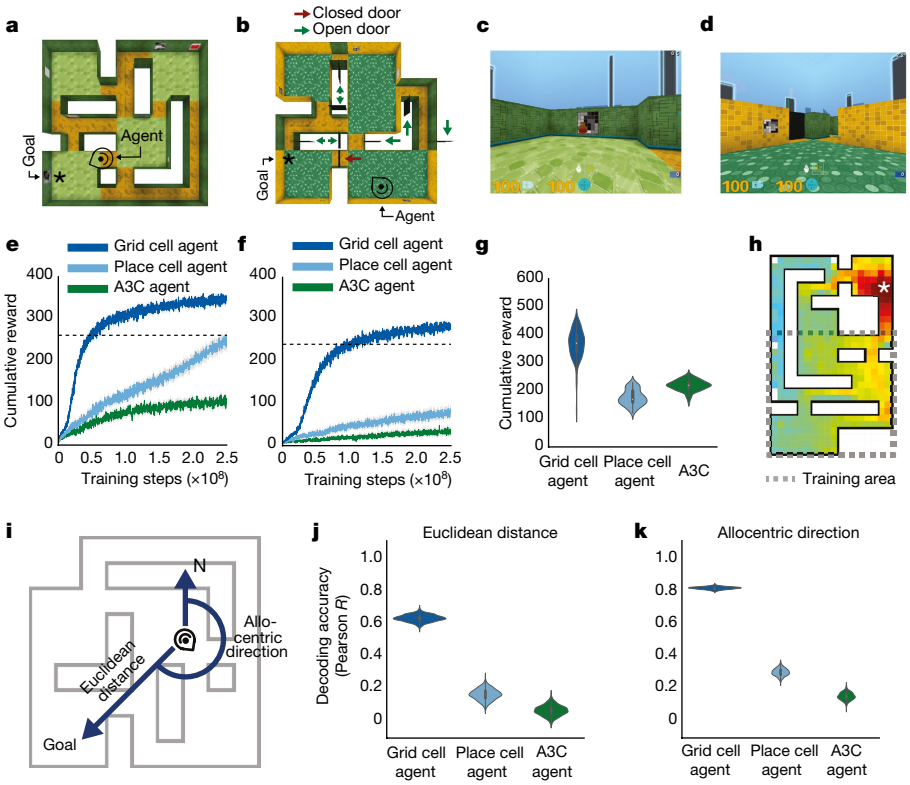


Fig. 3 | Navigation in challenging environments. a, **b**, Overhead view of multi-room environments goal-driven (**a**) and goal-doors (that is, with stochastic doors; **b**). Goal (asterisk) and agent (head icon) locations are displayed. **c**, **d**, Agent views of **a** and **b**, respectively, showing red goal and closed black door. **e**, **f**, Agent training performance curves for **a** and **b**, respectively, and performance of human expert (dashed line). Performance is mean cumulative reward over 100 episodes. The grey band displays the 68% CI based on 5,000 bootstrapped samples. **g**, Distribution of test performance over 100 episodes, showing ability of agents to generalize to a larger version of the goal-doors environment, displayed with a violin plot overlaid on a Tukey boxplot for each agent. **h**, The value function of the grid cell agent is projected onto an example larger goal doors environment as a heat map (where red indicates higher value function). Dotted lines show the extent of the original training environment. Despite the larger size, the value function clearly approximates Euclidean distance to goal. **i**, Schematic displaying the key metrics required for vector-based navigation to a goal. **j**, **k**, Decoding of vector-based metric codes from the policy LSTMs of agents during navigation. The bootstrapped distribution (1,000 samples) of correlation coefficients are displayed with a violin plot overlaid on a Tukey boxplot in each case.

reinforcement learning agents in game playing scenarios[2] (Fig. 3e, f; see Supplementary Results). Furthermore, when agents were tested without retraining in environments considerably larger than those seen previously, only the grid cell agent was able to generalize effectively (Fig. 3g, h; see Supplementary Results). Despite the complexity of the 'goal-driven' environment, we could still decode the key metric codes from the grid agent policy LSTM with high accuracy during the initial period of navigation, and decoding accuracy was substantially higher in the grid cell agent than in both the place cell and deep reinforcement learning control agents (Fig. 3j, k and Supplementary Results; see Extended Data Figs. 8, 9 for control agent architectures).

Finally, a core feature of mammalian spatial behaviour is the ability to exploit novel shortcuts and traverse unvisited portions of space, a capacity thought to depend on vector-based navigation[9,11]. Strikingly, the grid cell agent—but not comparison agents—robustly demonstrated these abilities in specifically designed neuroscience-inspired mazes, taking direct routes to the goal as soon as they became available (Fig. 4, Extended Data Fig. 10 and Supplementary Results).

Conventional simultaneous localization and mapping (SLAM) techniques typically require an accurate and complete map to be built, with the nature and position of the goal externally defined[30]. By contrast, the deep reinforcement learning approach described in this work has the ability to learn complex control policies end-to-end from a sparse reward, taking direct routes involving shortcuts to goals in an automatic fashion—abilities that exceed previous deep reinforcement learning approaches[3–5], and that would have to be hand-coded in any SLAM system.

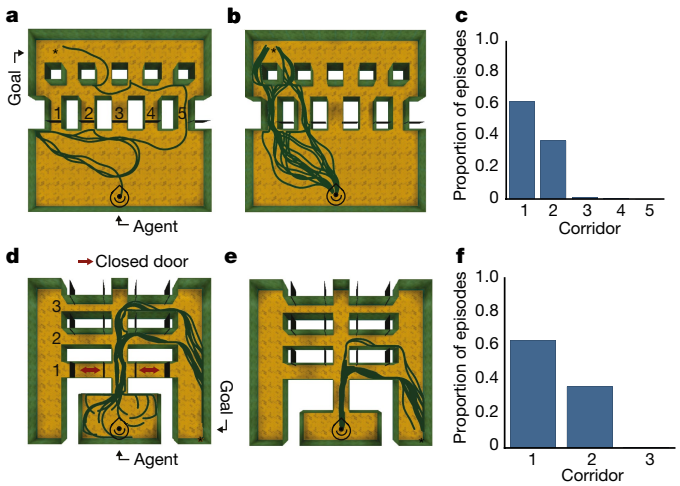


Fig. 4 | Flexible use of shortcuts. a, Example trajectory from grid cell agent in the linear sunburst maze (only door 5 open; icon indicates start location). **b**, Testing configuration with all doors open; grid cell agent uses the newly available shortcuts (100 episodes shown). **c**, Histogram showing agent's strong preference for most direct routes. **d**, Example grid cell agent trajectories (100) during training in the double E-maze (corridor 1 doors closed). **e**, Testing configuration with corridor 1 open, and 100 grid agent trajectories. **f**, Histogram analogous to **c** showing that agent prefers newly available shortest route. See Extended Data Fig. 10 for performance of place cell agent.

Our work, in demonstrating that grid-like representations provide an effective basis for flexible navigation in challenging novel environments, supports theoretical models of grid cells in vector-based navigation that were previously lacking strong empirical support[7,10,11]. We also show that vector-based navigation can be effectively combined with a path-based barrier avoidance strategy to enable the exploitation of optimal routes in challenging multi-compartment environments. In sum, we argue that grid-like representations furnish agents with a Euclidean geometric framework—paralleling their proposed computational role in mammals as an early-developing Kantian-like spatial scaffold that serves to organize perceptual experience[17,18].

Online content

Any Methods, including any statements of data availability and Nature Research reporting summaries, along with any additional references and Source Data files, are available in the online version of the paper at https://doi.org/10.1038/s41586-018-0102-6.

Received: 5 July 2017; Accepted: 3 April 2018;
Published online 9 May 2018.

1. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
2. Silver, D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
3. Oh, J., Chockalingam, V., Singh, S. P. & Lee, H. Control of memory, active perception, and action in Minecraft. *Proc. Intl Conf. Machine Learning* **48** (2016).
4. Kulkarni, T. D., Saeedi, A., Gautam, S. & Gershman, S. J. Deep successor reinforcement learning. Preprint at https://arxiv.org/abs/1606.02396 (2016).
5. Mirowski, P. et al. Learning to navigate in complex environments. *Intl Conf. Learning Representations* (2017).
6. Hafting, T., Fyhn, M., Molden, S., Moser, M.-B. & Moser, E. I. Microstructure of a spatial map in the entorhinal cortex. *Nature* **436**, 801–806 (2005).
7. Fiete, I. R., Burak, Y. & Brookings, T. What grid cells convey about rat location. *J. Neurosci.* **28**, 6858–6871 (2008).
8. Mathis, A., Herz, A. V. & Stemmler, M. Optimal population codes for space: grid cells outperform place cells. *Neural Comput.* **24**, 2280–2317 (2012).
9. McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I. & Moser, M.-B. Path integration and the neural basis of the 'cognitive map'. *Nat. Rev. Neurosci.* **7**, 663–678 (2006).
10. Erdem, U. M. & Hasselmo, M. A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *Eur. J. Neurosci.* **35**, 916–931 (2012).
11. Bush, D., Barry, C., Manson, D. & Burgess, N. Using grid cells for navigation. *Neuron* **87**, 507–520 (2015).
12. Barry, C. & Burgess, N. Neural mechanisms of self-location. *Curr. Biol.* **24**, R330–R339 (2014).
13. Mittelstaedt, M.-L. & Mittelstaedt, H. Homing by path integration in a mammal. *Naturwissenschaften* **67**, 566–567 (1980).
14. Bassett, J. P. & Taube, J. S. Neural correlates for angular head velocity in the rat dorsal tegmental nucleus. *J. Neurosci.* **21**, 5740–5751 (2001).
15. Kropff, E., Carmichael, J. E., Moser, M.-B. & Moser, E. I. Speed cells in the medial entorhinal cortex. *Nature* **523**, 419–424 (2015).
16. Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
17. Wills, T. J., Cacucci, F., Burgess, N. & O'Keefe, J. Development of the hippocampal cognitive map in preweanling rats. *Science* **328**, 1573–1576 (2010).
18. Langston, R. F. et al. Development of the spatial representation system in the rat. *Science* **328**, 1576–1580 (2010).
19. Zhang, S.-J. et al. Optogenetic dissection of entorhinal-hippocampal functional connectivity. *Science* **340**, 1232627 (2013).
20. Sargolini, F. et al. Conjunctive representation of position, direction, and velocity in entorhinal cortex. *Science* **312**, 758–762 (2006).
21. Barry, C., Hayman, R., Burgess, N. & Jeffery, K. J. Experience-dependent rescaling of entorhinal grids. *Nat. Neurosci.* **10**, 682–684 (2007).
22. Stensola, H. et al. The entorhinal grid map is discretized. *Nature* **492**, 72–78 (2012).
23. Stemmler, M., Mathis, A. & Herz, A. V. Connecting multiple spatial scales to decode the population activity of grid cells. *Sci. Adv.* **1**, e1500816 (2015).
24. Doeller, C. F., Barry, C. & Burgess, N. Evidence for grid cells in a human memory network. *Nature* **463**, 657–661 (2010).
25. Kanitscheider, I. & Fiete, I. Training recurrent networks to generate hypotheses about how the brain solves hard navigation problems. Preprint at https://arxiv.org/abs/1609.09059 (2016).
26. Milford, M. J. & Wyeth, G. F. Mapping a suburb with a single camera using a biologically inspired slam system. *IEEE Trans. Robot.* **24**, 1038–1053 (2008).
27. Hardcastle, K., Ganguli, S. & Giocomo, L. M. Environmental boundaries as an error correction mechanism for grid cells. *Neuron* **86**, 827–839 (2015).
28. Chen, G., King, J. A., Burgess, N. & O'Keefe, J. How vision and movement combine in the hippocampal place code. *Proc. Natl Acad. Sci. USA* **110**, 378–383 (2013).
29. Sarel, A., Finkelstein, A., Las, L. & Ulanovsky, N. Vectorial representation of spatial goals in the hippocampus of bats. *Science* **355**, 176–180 (2017).
30. Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H. F. & Csorba, M. A solution to the simultaneous localization and map building (slam) problem. *IEEE Trans. Robot. Autom.* **17**, 229–241 (2001).

Acknowledgements We thank M. Jaderberg, V. Mnih, A. Santoro, T. Schaul, K. Stachenfeld and J. Yosinski for discussions, and M. Botvinick and J. Wang for comments on an earlier version of the manuscript. C.Ba. funded by Royal Society and Wellcome Trust.

Reviewer information *Nature* thanks J. Conradt and the other anonymous reviewer(s) for their contribution to the peer review of this work.

Author contributions Conceived project: A.B., D.K., C.Ba., R.H., P.M. and B.U.; contributed ideas to experiments: A.B., D.K., C.Ba., B.U., R.H., T.L., C.Bl., P.M., A.P., T.D., J.M., K.K., N.R., G.W., R.G., M.J.C., D.H. and R.P.; performed experiments and analysis: A.B., C.Ba., B.U., M.J.C., T.L., H.S., A.P., B.Z. and F.V.; development of testing platform and environments: C.Be., S.P., R.H., T.L., G.W., D.K., A.B., B.U. and D.H.; human expert tester: A.S.; managed project: D.K., R.H., A.B., H.K., S.G. and D.H.; wrote paper; D.K., A.B., C.Ba., T.L., C.Bl., B.U., M.C., A.P., R.H., N.R., K.K. and D.H.

Competing interests The authors declare no competing interests.

Additional information
Extended data is available for this paper at https://doi.org/10.1038/s41586-018-0102-6.
Supplementary information is available for this paper at https://doi.org/10.1038/s41586-018-0102-6.
Reprints and permissions information is available at http://www.nature.com/reprints.
Correspondence and requests for materials should be addressed to A.B. or C.B. or D.K.
Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

METHODS

Path integration: supervised learning experiments. *Simplified 2D environment.* Simulated rat trajectories of duration T were generated in square and circular environments with walls of length L (diameter in the circular case). The simulated rat started at a uniformly sampled location and facing angle within the enclosure. A rat-like motion model[31] was used to obtain trajectories that uniformly covered the whole environment by avoiding walls (see Supplementary Methods Table 1 for the model's parameters).

Ground truth place cell distribution. Place cell activations, $\vec{c} \in [0,1]^N$, for a given position $\vec{x} \in R^2$ were simulated by the posterior probability of each component of a mixture of two- dimensional isotropic Gaussians,

$$c_i = \frac{e^{-\frac{\|\vec{x}-\vec{\mu}_i^{(c)}\|_2^2}{2(\sigma^{(c)})^2}}}{\sum_{j=1}^{N} e^{-\frac{\|\vec{x}-\vec{\mu}_j^{(c)}\|_2^2}{2(\sigma^{(c)})^2}}}$$

where $\vec{\mu}_i^{(c)} \in \mathbb{R}^2$, the place cell centres, are N two-dimensional vectors chosen uniformly at random before training, and $\sigma^{(c)}$, the place cell scale, is a positive scalar fixed for each experiment.

Ground truth head-direction cell distribution. Head-direction cell activations, $\vec{h} \in [0,1]^M$, for a given facing angle φ were represented by the posterior probability of each component of a mixture of Von Mises distributions with concentration parameter $\kappa^{(h)}$,

$$h_i = \frac{e^{k^{(h)}\cos(\varphi-\mu_i^{(h)})}}{\sum_{j=1}^{M} e^{k^{(h)}\cos(\varphi-\mu_j^{(h)})}}$$

where the M head direction centres $\mu_i^{(h)} \in [-\pi, \pi]$ are chosen uniformly at random before training, and $\kappa^{(h)}$, the concentration parameter, is a positive scalar fixed for each experiment.

Supervised learning inputs. In the supervised setup the grid cell network receives, at each step t, the egocentric linear velocity $v_t \in \mathbb{R}$ and the sine and cosine of its angular velocity φ_t.

Grid cell network architecture. The grid cell network architecture (Extended Data Fig. 1) consists of three layers: a recurrent layer, a linear layer, and an output layer. The single recurrent layer is an LSTM (long short-term memory[32]) that projects to place and head direction units via the linear layer. The linear layer implements regularization through dropout[16]. The recurrent LSTM layer consists of one cell of 128 hidden units, with no peephole connections. Input to the recurrent LSTM layer is the vector $[v_t, sin(\varphi_t), cos(\varphi_t)]$. The initial cell state and hidden state of the LSTM, $\vec{l}_0$ and $\vec{m}_0$, respectively, are initialized by computing a linear transformation of the ground truth place and head-direction cells at time 0,

$$\vec{l}_0 = W^{cp}\vec{c}_0 + W^{(cd)}\vec{h}_0$$
$$\vec{m}_0 = W^{hp}\vec{c}_0 + W^{(hd)}\vec{h}_0$$

The parameters of these two linear transformations ($W^{(cp)}$, $W^{(cd)}$, $W^{(hp)}$ and $W^{(hd)}$) were optimized during training. The output of the LSTM, $\vec{m}_t$ is then used to produce predictions of the place cells $\vec{y}_t$ and head direction cells $\vec{z}_t$ by means of a linear decoder network.

The linear decoder consists of three sets of weights and biases. The first set consists of the weights and biases that map from the LSTM hidden state $\vec{m}_t$ to the linear layer activations $\vec{g}_t \in \mathbb{R}^{512}$. The other two sets of weights map from the linear layer activations $\vec{g}_t$ to the predicted head directions, $\vec{z}_t$ and predicted place cells, $\vec{y}_t$, respectively, via softmax functions[33]. Dropout[16] with drop probability 0.5 was applied to each $\vec{g}_t$ unit. Note that there is no intermediary nonlinearity in the linear decoder.

Supervised learning loss. The grid cell network is trained to predict the place and head-direction cell ensemble activations, $\vec{c}_t$ and $\vec{h}_t$, respectively, at each time step t. During training, the network was trained in a single environment where the place cell centres were constant throughout. The parameters of the grid cell network are trained by minimizing the cross-entropy between the network place cell predictions, $\vec{y}_t$, and the synthetic place-cells targets, $\vec{c}_t$, and the cross-entropy between head-direction predictions, $\vec{z}_t$, and their targets, $\vec{h}_t$,

$$\mathcal{L}(\vec{y}, \vec{z}, \vec{c}, \vec{h}) = -\sum_{i=1}^{N} c_i \log(y_i) - \sum_{j=1}^{N} h_j \log(z_j)$$

Gradients of this loss function with respect to the network parameters were calculated using backpropagation through time[34], unrolling the network into blocks of 100 time steps. The network parameters were updated using stochastic-gradient descent (RMSProp[35]), with weight decay[36] for the weights projecting from the dropout linear layer, $\vec{g}_t$, to the place and head-direction cell predictions, $\vec{y}_t$ and $\vec{z}_t$. Hyperparameter values used for training are listed in Supplementary Methods Table 1.

Gradient clipping. In our simulations, gradient clipping was used for parameters projecting from the dropout linear layer, $\vec{g}_t$, to the place and head-direction cell predictions $\vec{y}_t$ and $\vec{z}_t$. Gradient clipping clips each element of the gradient vector to lie in a given interval $[-g_c, g_c]$, and is an important tool for optimization in deep and recurrent artificial neural networks, where it helps by preventing exploding gradients[37]. Gradient clipping also introduces distortions into the weight updates, which help to avoid local minima[38].

Navigation through deep reinforcement learning. *Environments and task.* We assessed the performance of agents on three environments seen by the agent from a first-person perspective in the DeepMind Laboratory[39] platform.

Custom environment: square arena. This comprised a 10×10 square arena, which corresponds to 2.5×2.5 m arena assuming an agent speed of 15 cm/s (Fig. 2b, c). The arena contained a single, coloured, intra-arena cue whose position and colour changed on each episode, as did the texture of the floor, the texture of the walls and the goal location. As in the goal-driven and goal-door environments described below, there were a set of distal cues (that is, buildings) that paralleled the design of virtual reality environments used in human experiments[40]. These distal cues were rendered at infinity—so as to provide directional but not distance information—and their configuration was consistent across episodes. At the start of each episode the agent (described below) started in a random location and was required to explore in order to find an unmarked goal, paralleling the task of rodents in the classic Morris water maze. The agent always started in the central 6×6 grids (that is, 1.5×1.5 m) of the environment. Noise in the velocity input $\vec{u}_t$ was applied throughout training and testing (that is, Gaussian noise ε, with $\mu = 0$ and $\sigma = 0.01$). The action space is discrete (six actions) but affords fine-grained motor control (that is, the agent could rotate in small increments, accelerate forwards, backwards or sideways, or effect rotational acceleration while moving).

DeepMind laboratory environments: goal-driven and goal-doors. Goal-driven and goal-doors are challenging, visually rich multi-room environments (Fig. 3a–d). Mazes were formed within an 11×11 grid, corresponding to 2.7×2.7 m (see below for definition of larger 11×17 mazes). Mazes were procedurally generated at the beginning of each episode; thus, the layout, wall texture, landmarks (intra-maze cues on walls) and goal location were different for each episode but consistent within an episode. Distal cues, in the form of buildings rendered at infinity, were as described above for the square arena.

The critical difference between goal-driven and goal-doors tasks is that the latter had the additional challenge of stochastic doors within the maze. Specifically, the state of the doors (open or closed) changed randomly during an episode each time the agent reached the goal. This meant that the optimal path to the goal from a given location changed during an episode, requiring the agent to recompute trajectories.

In both tasks, the agent starts at a random location within the maze and its task is to explore to find the goal. The goal in both levels was always represented by the same object (Fig. 3c). After getting to the goal the agent received a reward of 10 points, after which it was teleported to a new random location within the maze. In both levels, episodes lasted a fixed duration of 5,400 environment steps (90 s).

Generalization on larger environments. We tested the ability of agents trained on the standard environment (11×11) to generalize to larger environments (11×17, corresponding to 2.7×4.25 m). The procedural generation and composition of these environments was done as for the standard environments. Each agent was trained in the 11×11 goal-doors maze for a total of 10^9 environment steps, and the best performing replica (highest asymptotic performance averaged over 100 episodes in 11×11) was selected for evaluation in the larger maze. Note that the weights of the agent were frozen during evaluation on the larger maze. Evaluation was over 100 episodes of fixed duration 12,600 environment steps (210 s).

Probe mazes to assess shortcut behaviour. To test the agent's ability to follow novel, goal- directed routes, we created a series of environments inspired by mazes designed to test the shortcut abilities of rodents.

The first maze is a linearized version of Tolman's sunburst maze (Fig. 4a) used to determine whether the agent was able to follow an accurate heading towards the goal when a path became available (see Supplementary Methods). In this maze, after reaching the goal, the agent was teleported to the original position with the same heading orientation. Agents were trained in the goal-doors maze and network weights were frozen during testing—all the agents were tested for 100 episodes, each one lasting for a fixed duration of 5,400 environment steps (90 s).

The second environment, the double E-maze (Fig. 4d, Extended Data Fig. 10), was designed to test the agent's ability to traverse an entirely new portion of space (see Supplementary Methods). In this maze we had both training and testing conditions. Both training and testing were conducted within the maze but at test time weights were frozen. The agent always started in the central room (for example,

see Fig. 4d). The maze had stochastic doors with two different configurations, one for the training phase and one for testing phase. During training, the state of the doors (open or closed) randomly changed during an episode each time the agent reached the goal. Critically, during training the corridors presenting the shortest route to the goal (that is, the ones closer to the central room) were closed at both ends, preventing access to or observation of the interior. At test time, after the agent had reached the goal for the first time, all doors were opened. All the agents were tested for 100 episodes, each one lasting for a fixed duration of 5,400 environment steps (90 s).

Agent architectures. *Architecture for the grid cell agent.* The agent architecture (Extended Data Fig. 5) was composed of a visual module, the grid cell network (described above), and an actor–critic learner[41]. The visual module was a neural network with input consisting of a three-channel (RGB) 64×64 image $\phi \in [-1, 1]^{3\times84\times84}$. The image was processed by a convolutional neural network (see Supplementary Methods for details), which produced embeddings, $\vec{e}$, which in turn were used as input to a fully connected linear layer trained in a supervised fashion to predict place and head-direction cell ensemble activations, $\vec{c}$ and $\vec{h}$ (as specified above), respectively. The predicted place and head direction cell activity patterns were provided as input to the grid network 5% of the time on average, akin to occasional imperfect observations made by behaving animals of salient environmental cues[27]. Specifically, the output of the convolutional network $\vec{e}$ was then passed through a masking layer which zeroed the units with a probability of 95%.

The grid cell network of the agent was implemented as in the supervised learning set up except that the LSTM ('grid LSTM') was not initialized with ground truth place cell activations but instead set to zero. The inputs to the grid cell network were the two translation velocities, u and v, as in DeepMind Laboratory it is possible to move in a direction different from the facing direction; the sine and cosine of the angular velocity, $\dot{\varphi}$, (these velocities are provided by DeepMind Laboratory); and the $\vec{y}$ and $\vec{z}$ output by the vision module. In contrast to the supervised learning case, here the grid cell network had to use $\vec{y}$ and $\vec{z}$ to learn how to reset its internal state each time it was teleported to an arbitrary location in the environment (for example, after visiting the goal). As in the supervised learning experiments described above, the configuration of place fields (that is, location of place field centres in the 11×11 environments, goal-driven and goal-doors, 10×10 square arena, and 13×13 double E-maze) were constant throughout training (that is, across episodes).

For the actor–critic learner, the input was a three-channel 64×64 image $\phi_t \in [-1, 1]^{3\times84\times84}$, which was processed by a convolutional neural network followed by a fully connected layer (see Supplementary Methods for details). The output of the fully connected layer of the convolutional network $\vec{e}_t$ was then concatenated with the reward r_t, the previous action a_{t-1}, the current grid code $\vec{g}_t$, and the goal grid code $\vec{g}_*$ (that is, linear layer activations observed last time the goal was reached)—or zeros if the goal had not yet been reached in the episode. Note we refer to these linear layer activations as 'grid codes' for brevity, even though units in this layer also comprise units resembling head direction cells and border cells (Extended Data Fig. 6a). This concatenated input was provided to an LSTM with 256 units. The LSTM had two different outputs. The first output, the actor, is a linear layer with six units followed by a softmax activation function, which represents a categorical distribution over the agent's next action. The second output, the critic, is a single linear unit that estimates the value function. Note that we refer to this as the 'policy LSTM' for brevity, even though it also outputs the value function.

Comparison agents. We compared the performance of the grid cell agent against two agents specifically because they use a different representational scheme for space (that is, place cell agent, place cell prediction agent), and relate to theoretical models of goal-directed navigation from the neuroscience literature (for example[42,43]). We also compared the grid cell agent against a baseline deep reinforcement learning agent, Asynchronous Advantage Actor–Critic (A3C)[41].

Place cell agent. The place cell agent architecture is shown in Extended Data Fig. 8b, and described in more detail in Supplementary Methods. In contrast to the grid cell agent, the place cell agent used ground truth information: specifically, the ground-truth place, $\vec{c}_t$, and head-direction, $\vec{h}_t$, cell activations (as described above). These activity vectors were provided as input to the policy LSTM in an analogous way to the provision of grid codes in the grid cell agent.

Specifically, the output of the fully connected layer of the convolutional network $\vec{e}_t$ was concatenated with the reward r_t, the previous action a_{t-1}, the ground-truth current place code, $\vec{c}_t$, and the current head-direction code, $\vec{h}_t$, together with the ground truth goal place code, $\vec{c}_*$, and ground truth head direction code, $\vec{h}_*$, observed last time the goal was reached—or with zeros if the goal had not yet been reached in the episode (Extended Data Fig. 8b). The convolutional network had the same architecture as described for the grid cell agent.

Place cell prediction agent. The architecture of the place cell prediction agent (Extended Data Fig. 9a) is similar to that of the grid cell agent described above: the key difference is the nature of the input provided to the policy LSTM as described below. The place cell prediction agent had a grid cell network with the same parameters as that of the grid cell agent. However, instead of using grid codes from the linear layer of the grid network $\vec{g}$ as input for the policy LSTM (as in the grid cell agent), we used the predicted place cell population activity vector $\vec{y}$ and the predicted head direction population activity vector $\vec{z}$ (the activations present on the output place and head direction unit layers of the grid cell network at each timestep; see Supplementary Methods).

The critical difference between the place cell agent and the place cell prediction agent (see Extended Data Figs. 8b and 9a, respectively) is that the former used ground truth information (place and head direction cell activations for current location and goal location), whereas the latter used the population activity produced across the output place and head direction cell layers (for current location and goal location) by the linear layer of the same grid network as used by the grid cell agent.

A3C. We implemented the asynchronous advantage actor–critic architecture[41] with a convolutional network having the same architecture as described for the grid cell agent (Extended Data Fig. 8a).

Other agents. We also assessed the performance of two deep reinforcement learning agents with external memory (Extended Data Fig. 9b), which served to establish the challenging nature of the multi-compartment environments (goal-doors and goal-driven). First, we implemented a memory network agent (NavMemNet) consisting of the FRMQN architecture[3] but instead of Q-learning we used the A3C algorithm described below. Furthermore, the input to memory was generated as an output from the LSTM controller (Extended Data Fig. 9b), rather than constituting embeddings from the convolutional network (that is, as in ref. 3). The convolutional network had the same architecture as described for the grid cell agent and the memory was formed of two banks (keys and values), each one with 1,350 slots.

Second, we implemented a differentiable neural computer (DNC) agent that uses content-based retrieval and writes to the most recently used or least recently used memory slot[44].

Training algorithms. We used the A3C algorithm[41], which implements a policy $\pi(a|s, \theta)$ and an approximation to its value function $V(s, \theta)$ using a neural network parameterised by θ. A3C adjusts the network parameters using n-step lookahead values, $\hat{R}_t = \sum_{i=0\ldots n-1} \gamma^i r_{t+1} + \gamma^n V(s_{t+n}, \theta)$, to minimize: $\mathcal{L}_{A3C} = \mathcal{L}_\pi + \alpha\mathcal{L}_V + \beta\mathcal{L}_H$, where $\mathcal{L}_\pi = -\mathbb{E}_{s_t\sim\pi}[\hat{R}_t]$, $\mathcal{L}_V = -\mathbb{E}_{s_t\sim\pi}[(\hat{R}_t - V(s_t, \theta))^2]$, $\mathcal{L}_\pi = -\mathbb{E}_{s_t\sim\pi}[H(\pi(\cdot|\, s_t, \theta))]$ where $\mathcal{L}_H$ is a policy entropy regularization term (see Supplementary Methods for details of the reinforcement learning approach). The grid cell network and the vision module were trained with the same loss reported for the supervised learning: $\mathcal{L}(\vec{y}, \vec{z}, \vec{c}, \vec{h}) = -\sum_{i=1}^{N} c_i \log(y_i) - \sum_{j=1}^{N} h_j \log(z_j)$.

Agent training details. We followed closely a previously described approach[41]. Each experiment used 32 actor–critic learner threads running on a single CPU machine. All threads applied updates to their gradients every four actions (that is, action repeat of 4) using RMSProp with shared gradient statistics[41]. All the experiments were run for a total of 10^9 environment steps.

In architectures where the grid cell network and the vision module were present, we used a shared buffer[45,46] in which we stored the agent's experiences at each time-step, $e_t = (\phi_t, u_t, v_t)$, collected over many episodes. All the 32 actor–critic workers were updating the same shared buffer which had a total size of 20×10^6. The vision module was trained with mini batches of size 32 frames ($\vec{\phi}$) sampled randomly from the replay buffer. The grid cell network was trained with mini batches of size 10, randomly sampled from the buffer, each one comprising a sequence of 100 consecutive observations, [$\vec{\phi}$, $\vec{u}$, $\vec{v}$]. These mini batches were first forwarded through the vision module to get $\vec{c}$, and $\vec{h}$, which were then passed through a masking layer that masked them to 0 with a probability of 95% (as described above in section on grid cell architecture). The output of this masking layer was then concatenated with $\vec{u}$, $\vec{v}$, $\overrightarrow{sin\dot{\varphi}}$, $\overrightarrow{cos\dot{\varphi}}$, which were then used as inputs to the grid network, as previously described (see Extended Data Fig. 5 for details). Both networks were trained using one single thread, one to train the vision module and another to train the grid network (so in total we used 34 threads). Also, there was no gradient sharing between the actor–critic learners, the vision module and the grid network.

The hyperparameters of the grid cell network were kept fixed across all the simulations and were derived from the best performing network in the supervised learning experiments. For the hyperparameter details of the vision module, the grid network and the actor–critic learner, see Supplementary Table 2 in Supplementary Methods. For each of the agents in this paper, 60 replicas were run with hyperparameters sampled from the same interval and different initial random seeds.

Details of lesion experiment. To conduct a lesioning experiment in the agent, we trained the grid cell agent with dropout applied on the goal grid code input $\vec{g}_t$. Specifically, every 100 training steps we generated a random mask to silence 20% of the units in the goal grid code ($\vec{g}_*$)—that is, units were zeroed. This procedure was implemented to ensure that the policy LSTM would become robust through

training to receiving a lesioned input (that is, would not catastrophically fail), and still be able to perform the task.

We then selected the agent with the best performance over 100 episodes, and we computed the grid score of all units found in $\vec{g}$. The critical comparison to test the importance of grid-like units to vector-based navigation was as follows. In one condition we ran 100 testing episodes where we silenced the 25% units in $\vec{g}_*$ with the highest grid scores. In the other condition, we ran 100 testing episodes with the same agent with 25% random units in $\vec{g}_*$ silenced. In this second case, we ensured that head direction cells with a resultant vector length of more than 0.47 were not silenced, to preserve crucial head direction signals. We then compared the performance, and the representation of metrics relating to vector-based navigation, of the agents under these two conditions.

Details of experiment using 'fake' goal grid code. To demonstrate that the goal grid code provided sufficient information to enable the agent to navigate to an arbitrary location, we took an agent trained in the square arena, froze the weights, and ran it in the same square arena for 5,400 steps. Critically, after the sixth time the agent reached the goal, we sampled the grid code from a random point that the agent visited in the environment (the fake goal grid code). We then substituted the true goal grid code with this fake goal grid code, to show that this would be sufficient to direct the agent to a location where there was no actual goal.

Agent performance. To evaluate agent performance during training (as in Figs. 2f, 3e, f) we selected the 30 replicas (out of 60) with the highest average cumulative reward across 100 episodes. We also assessed the robustness of the architecture over different initial random seeds and the hyperparameters in Supplementary Methods Table 2 by calculating the area under the curve (AUC). To plot the AUC we ran 60 replicas with hyperparameters sampled from the same interval (see Supplementary Table 2 in Supplementary Methods) and different initial random seeds (Extended Data Fig. 7a–c).

Neuroscience-based analyses of network units. *Generation of activity maps.* Spatial (ratemaps) and directional activity maps were calculated for individual units as follows. Each point in the trajectory was assigned to a specific spatial and directional bin according to its location and the direction in which it faced. Spatial bins were defined as a 32×32 square grid spanning each environment and directional bins as 20 equal width intervals. Then, for each unit, the mean activity over all the trajectories points assigned to that bin was found. These values were displayed and analysed further without additional smoothing.

Inter-trial stability. For each unit, the reliability of spatial firing between baseline trials was assessed by calculating the spatial correlation between pairs of rate maps taken at two different logging steps in training ($t = 2 \times 10^5$; $t' = 3 \times 10^5$). The total training time was 3×10^5, so the points were selected with enough time difference to minimize the chances of finding random correlations. The Pearson product moment correlation coefficient was calculated between equivalent bins in the two trials and unvisited bins were excluded from the measure.

Quantification of spatial activity. Where possible, we assessed the spatial modulation of units using measures adopted from the neuroscience literature. The hexagonal regularity and scale of grid-like patterns were quantified using the gridness score[18,20] and grid scale[20], measures derived from the spatial autocorellogram[20] of each unit's ratemap. Similarly, the degree of directional modulation exhibited by each unit was assessed using the length of the resultant vector[47] of the directional activity map. Finally, the propensity of units to fire along the boundaries of the environment was quantified using the border score[48].

The gridness and border scores exhibited by units in the linear layer were benchmarked against the 95th percentile of null distributions obtained using a permutation procedure (spatial field shuffle[49]) applied to each unit's ratemap. This shuffling procedure aimed to preserve the local topography of fields within each ratemap while distributing the fields themselves at random[49]. The means, over units, of the thresholds obtained were gridness >0.37 and border score >0.50. Units exceeding these thresholds were considered to be grid-like and border-like, respectively. To identify directionally modulated cells, we applied Rayleigh tests of directional uniformity to the binned directional activity maps. A unit was considered to be directionally modulated if the null hypothesis of uniform was rejected at the $\alpha = 0.01$ level, corresponding to units with resultant vector length in excess of 0.47 (see Supplementary Methods).

Clustering of scale in grid-like units. To determine whether grid-like units exhibited a tendency to cluster around specific scales, we applied two methods. First, to determine whether the scales of grid-like units (gridness > 0.37, 129/512 units) followed a continuous or discrete distribution, we calculated the discreteness measure[22] of the distribution of their scales (see Supplementary Methods). The discreteness score of the real data was found to exceed that of all of the 500 shuffles. Second, to characterize the number and location of scale clusters, the distribution of scales from grid-like units was fit with Gaussian mixture distributions, and three components were found to provide the most parsimonious fit, indicating the presence of three scale clusters (See Supplementary Methods).

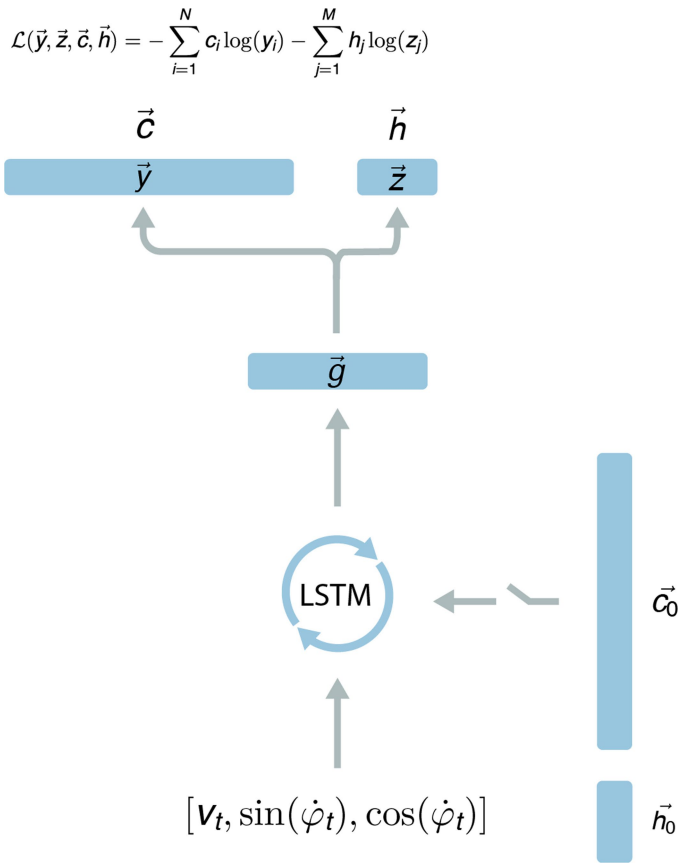
Multivariate decoding of representation of metric quantities within LSTM. To test whether the grid agent learns to use the predicted vector based navigation (VBN) metric codes, we recorded the activation from the hidden units of the policy LSTM layer while the agent navigated 200 episodes in the land maze. We used L2-regularized (ridge) regression to decode Euclidean distance and allocentric direction to the goal (see Supplementary Methods for full decoding details). We specifically focused on twelve steps (steps 9–21) during the early portion of navigation, but after the agent has had time to accurately self-localize. It is this early period after the agent has reached the goal for the first time in which a VBN strategy should be most effective. We conducted the same analysis on the place cell agent control, which is not predicted to use vector-based navigation as efficiently. The decoding accuracy was measured as the correlation between predicted and actual metric values in held-out data. Decoding accuracy was compared across different agents by assessing the difference in decoding correlations between the agents. A bootstrap method (using 10,000 samples) was used to compute a 95% confidence interval on this correlation difference, and these are reported for each comparison. The same approach was used to decode and compare these two metrics in the lesioned grid agents on the land maze. Finally, to explore VBN metrics in a more complex environment, the same method was applied to the goal-driven task. In this case we also investigated metric decoding in the control A3C agent.

Reporting summary. Further information on experimental design is available in the Nature Research Reporting Summary linked to this paper.

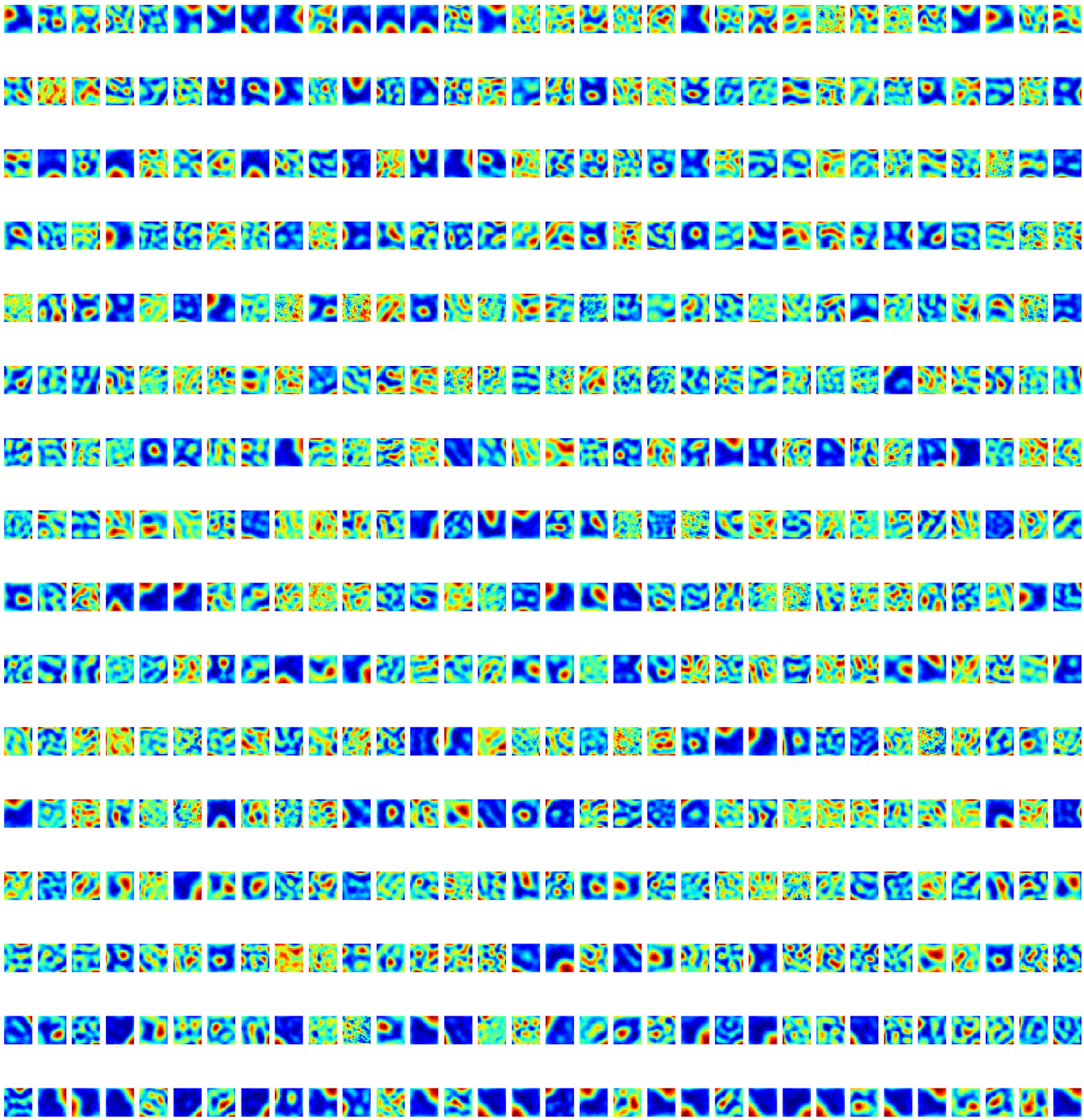
Data availability statement. All reinforcement learning tasks described throughout the paper were built using the publicly available DeepMind Laboratory platform (https://github.com/deepmind/lab). Both the goal driven and goal doors tasks are included as part of the latest release, named explore_goal_locations, and explore_obstructed_goals, respectively.

Code availability statement. We will release the code for the supervised learning experiments within the next six months. The codebase for the deep reinforcement learning agents makes use of proprietary components, and we are unable to publicly release this code. However, all experiments and agents are described in sufficient detail to allow independent replication.

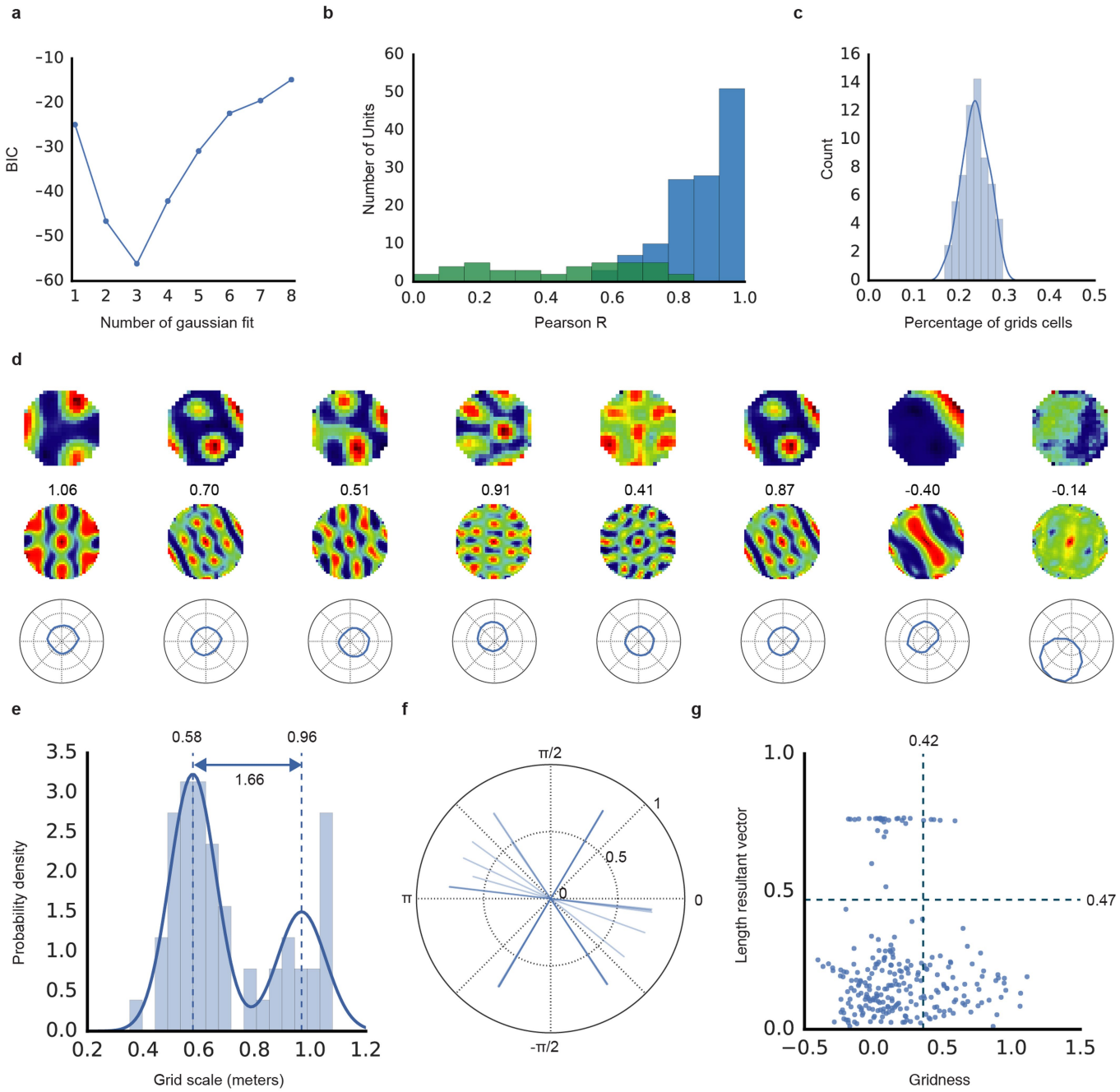
31. Raudies, F. & Hasselmo, M. E. Modeling boundary vector cell firing given optic flow as a cue. *PLOS Comput. Biol.* **8**, e1002553 (2012).
32. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
33. Bridle, J. S. in Touretzky, D. S. (ed.) *Advances in Neural Information Processing Systems 2* 211–217 (Morgan-Kaufmann, 1990).
34. Elman, J. L. & McClelland, J. L. Exploiting lawful variability in the speech wave. *Invariance and Variability in Speech Processes* **1**, 360–380 (1986).
35. Tieleman, T. & Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning (2012).
36. MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural Comput.* **4**, 448–472 (1992).
37. Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training recurrent neural networks. Proc. 30th ICML 28, 1310–1318 (2013).
38. Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. A learning algorithm for Boltzmann machines. *Cogn. Sci.* **9**, 147–169 (1985).
39. Beattie, C. et al. Deepmind lab. Preprint at https://arxiv.org/abs/1612.03801 (2016).
40. Doeller, C. F., Barry, C. & Burgess, N. Evidence for grid cells in a human memory network. *Nature* **463**, 657–661 (2010).
41. Mnih, V. et al. Asynchronous methods for deep reinforcement learning. In *Proc. 33nd Intl Conf. Machine Learning* 1928–1937 (2016).
42. Touretzky, D. S. & Redish, A. D. Theory of rodent navigation based on interacting representations of space. *Hippocampus* **6**, 247–270 (1996).
43. Foster, D. J., Morris, R. G. & Dayan, P. A model of hippocampally dependent navigation, using the temporal difference learning rule. *Hippocampus* **10**, 1–16 (2000).
44. Graves, A. et al. Hybrid computing using a neural network with dynamic external memory. *Nature* **538**, 471–476 (2016).
45. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
46. Lin, L.-J. Reinforcement learning for robots using neural networks. Technical Report (Carnegie-Mellon Univ. School of Computer Science, 1993).
47. Knight, R. et al. Weighted cue integration in the rodent head direction system. *Phil. Trans. R. Soc. Lond. B* **369**, 20120512 (2013).
48. Solstad, T., Boccara, C. N., Kropff, E., Moser, M.-B. & Moser, E. I. Representation of geometric borders in the entorhinal cortex. *Science* **322**, 1865–1868 (2008).
49. Barry, C. & Burgess, N. To be a grid cell: Shuffling procedures for determining gridness. Preprint at https://www.biorxiv.org/content/early/2017/12/08/230250 (2017).



Extended Data Fig. 1 | Network architecture in the supervised learning experiment. The recurrent layer of the grid cell network is an LSTM with 128 hidden units. The recurrent layer receives as input the vector $[\vec{v}, \sin(\mathring{\varphi}), \cos(\mathring{\varphi})]$. The initial cell state and hidden state of the LSTM, $\vec{l}_0$ and $\vec{m}_0$, respectively, are initialized by computing a linear transformation of the ground truth place $\vec{c}_0$ and head-direction activity $\vec{h}_0$ at time 0. The output of the LSTM is followed by a linear layer on which dropout is applied. The output of the linear layer, $\vec{g}_t$, is linearly transformed and passed to two softmax functions that calculate the predicted head direction cell activity, $\vec{z}_t$, and place cell activity, $\vec{y}_t$. We found evidence of grid-like and head direction-like units in the linear layer activations $\vec{g}_t$.

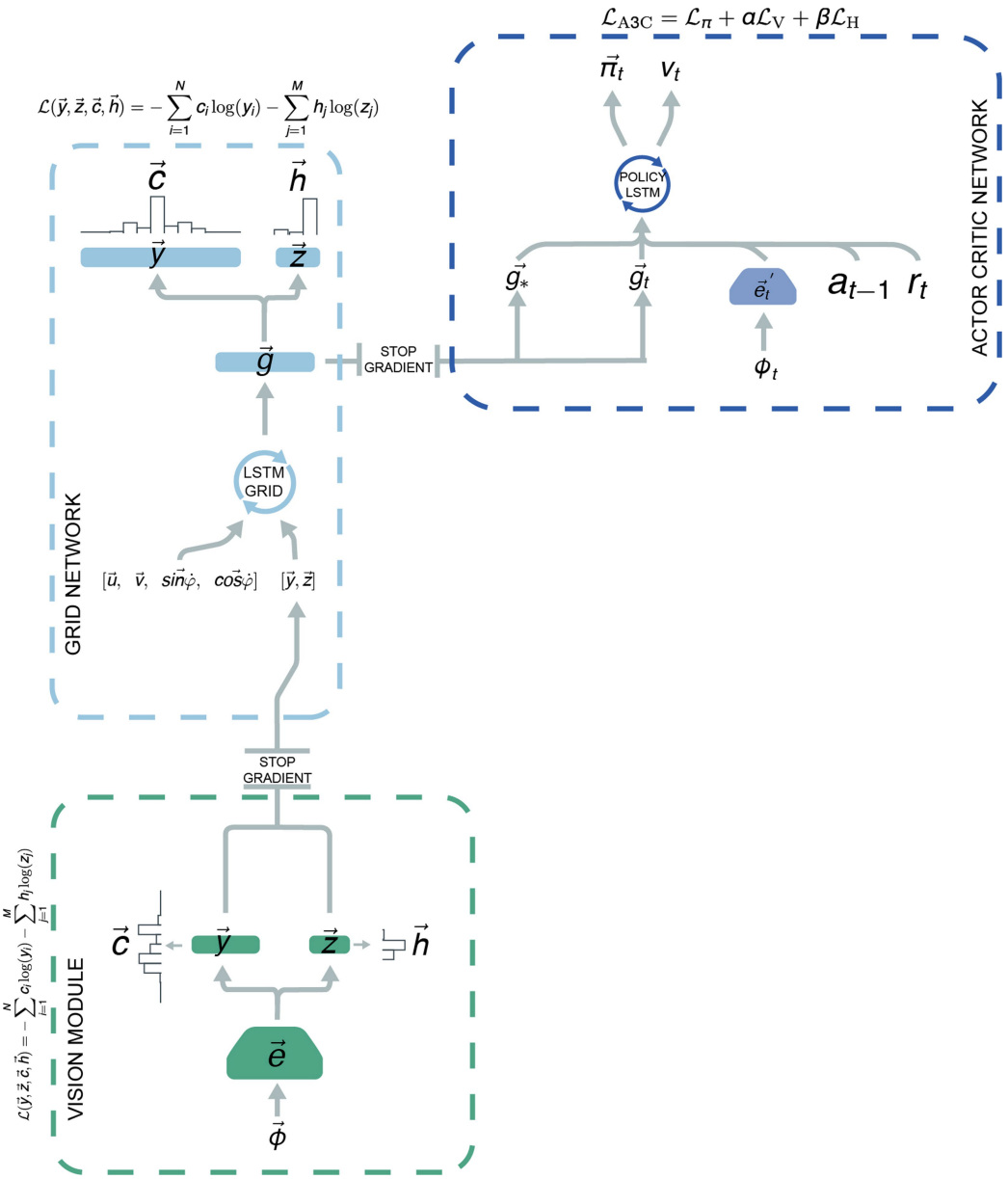


Extended Data Fig. 2 | Linear layer spatial activity maps from the supervised learning experiment. Spatial activity plots for all 512 units in the linear layer $\overrightarrow{g_t}$. Units exhibit spatial activity patterns resembling grid cells, border cells, and place cells. Head direction tuning was also present but is not shown.

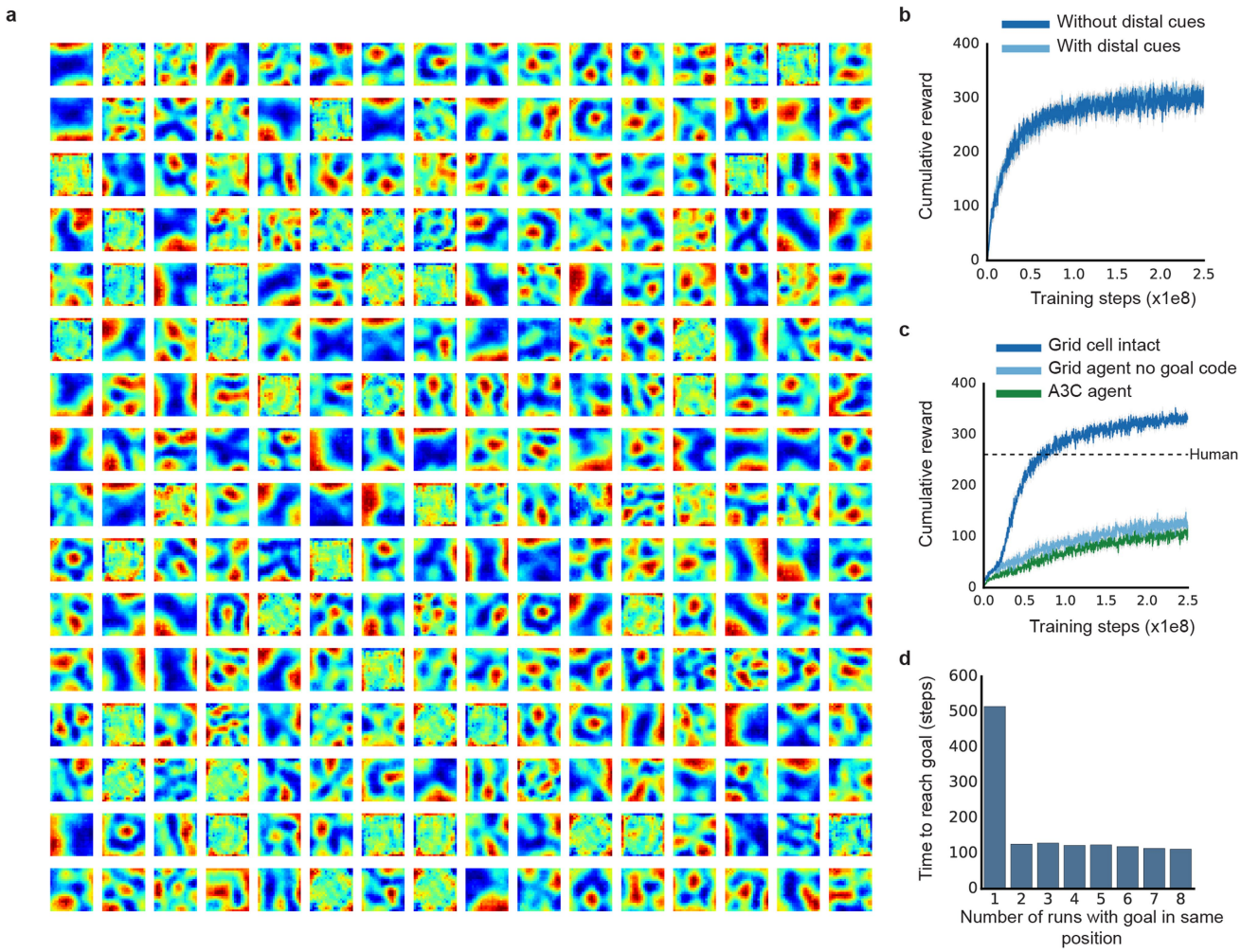


Extended Data Fig. 3 | Characterization of grid-like units in square environment and circular environment. a, The scale (assessed from the spatial autocorrelogram of the ratemaps) of grid-like units exhibited a tendency to cluster at specific values. The number of distinct scale clusters was assessed by sequentially fitting Gaussian mixture models with one to eight components. In each case, the efficiency of the fit (likelihood versus number of parameters) was assessed using Bayesian information criterion (BIC). BIC was minimized with three Gaussian components, indicating the presence of three distinct scale clusters. **b**, Spatial stability of units in the linear layer of the supervised network was assessed using spatial correlations— bin-wise Pearson product moment correlation between spatial activity maps (32 spatial bins in each map) generated at two different points in training, $t = 2 \times 10^5$ and $t' = 3 \times 10^5$ training steps (two-thirds of the way through training and at the end of training, respectively). This separation was imposed to minimize the effect of temporal correlations and to provide a conservative test of stability. Grid-like units (gridness > 0.37), blue; directionally modulated units (resultant vector length > 0.47, green. Grid-like units exhibit high spatial stability, while directionally modulated units do not. **c**, Robustness of the grid representation to starting conditions. The network was retrained 100 times with the same hyperparameters but different random seeds controlling the initialization of network weights, $\overrightarrow{c}$ and $\overrightarrow{h}$. Populations of grid-like units (gridness > 0.37) were found to appear in all cases, with the average proportion of grid-like units being 23% (s.d. 2.8%). **d**, The supervised network was also trained in a circular environment (diameter 2.2 m). As before, units in the linear layer exhibited spatially tuned responses resembling grid, border, and head direction cells. Eight units are shown. Top, ratemap displaying activity binned over location. Middle, spatial autocorrelogram of the ratemap; gridness[20] is indicated above. Bottom, polar plot of activity binned over head direction. **e**, Spatial scale of grid-like units ($n = 56$ (21.9%)) is clustered. Distribution is best fit by a mixture of two Gaussians (centres 0.58 and 0.96 m, ratio 1.66). **f**, Distribution of directional tuning for 31 most directionally active units; single line for each unit indicates length and orientation of resultant vector[47]. **g**, Distribution of gridness and directional tuning. Dashed lines indicate 95% confidence interval derived from shuffling procedure (500 permutations); five grid units (9%) exhibit significant directional modulation.

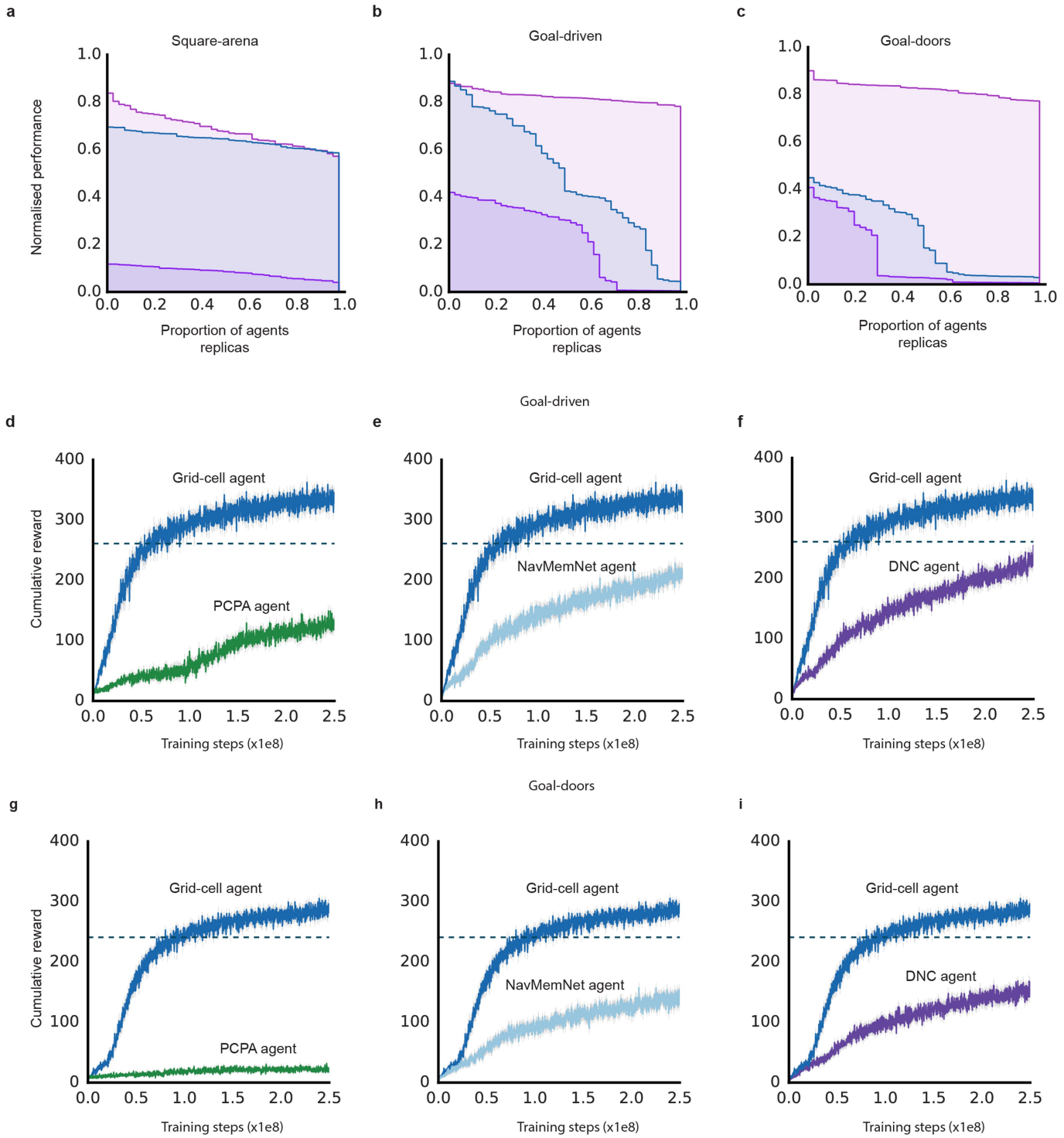
Extended Data Fig. 4 | Grid-like units did not emerge in the linear layer when dropout was not applied. Linear layer spatial activity maps ($n = 512$) generated from a supervised network trained without dropout. The maps do not exhibit the regular periodic structure diagnostic of grid cells.



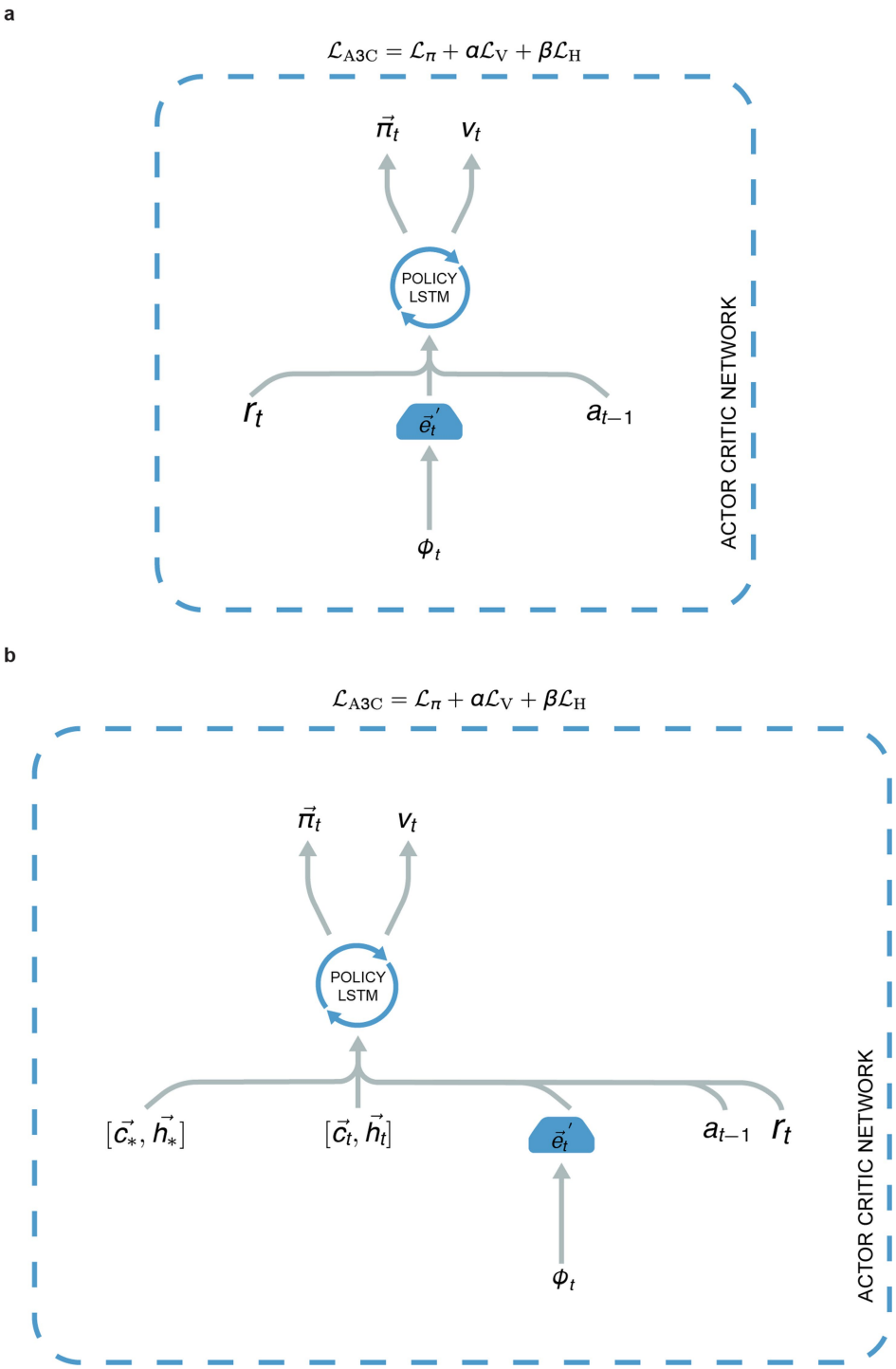
Extended Data Fig. 5 | Architecture of the grid cell agent. The architecture of the supervised network (grid network, light blue dashed) was incorporated into a larger deep reinforcement learning network, including a visual module (green dashed) and an actor–critic learner (based on A3C[41]; dark blue dashed). In this case the supervised learner does not receive the ground truth $\vec{c}_0$ and $\vec{h}_0$ to signal its initial position, but uses input from the visual module to self-localize after placement at a random position within the environment. Visual module: since experimental evidence suggests that place cell input to grid cells functions to correct for drift and anchor grids to environmental cues[21,27], visual input was processed by a convolutional network to produce place cell (and head direction cell) activity patterns which were used as input to the grid network. The output of the vision module was only provided 5% of the time to the grid network (see Methods for implementational details), akin to occasional observations of salient environmental cues made by behaving animals[27]. The output of the vision module was concatenated with $\vec{u}$, $\vec{v}$, $sin\vec{\varphi}$, $cos\vec{\varphi}$ to form the input to the grid LSTM, which is the same network as in the supervised case (see Methods and Extended Data Fig. 1). The actor–critic learner (light blue dashed) receives as input the concatenation of $\vec{e}_t'$ produced by a convolutional network with the reward r_t, the previous action a_{t-1}, the linear layer activations of the grid cell network $\vec{g}_t$ (current grid-code), and the linear layer activations observed last time the goal was reached, $\vec{g}_*$ (goal grid-code), which is set to zero if the goal has not been reached in the episode. The fully connected layer was followed by an LSTM with 256 units. The LSTM has two different outputs. The first output, the actor, is a linear layer with six units followed by a softmax activation function, which represents a categorical distribution over the agent's next action $\vec{\pi}_t$. The second output, the critic, is a single linear unit that estimates the value function v_t.



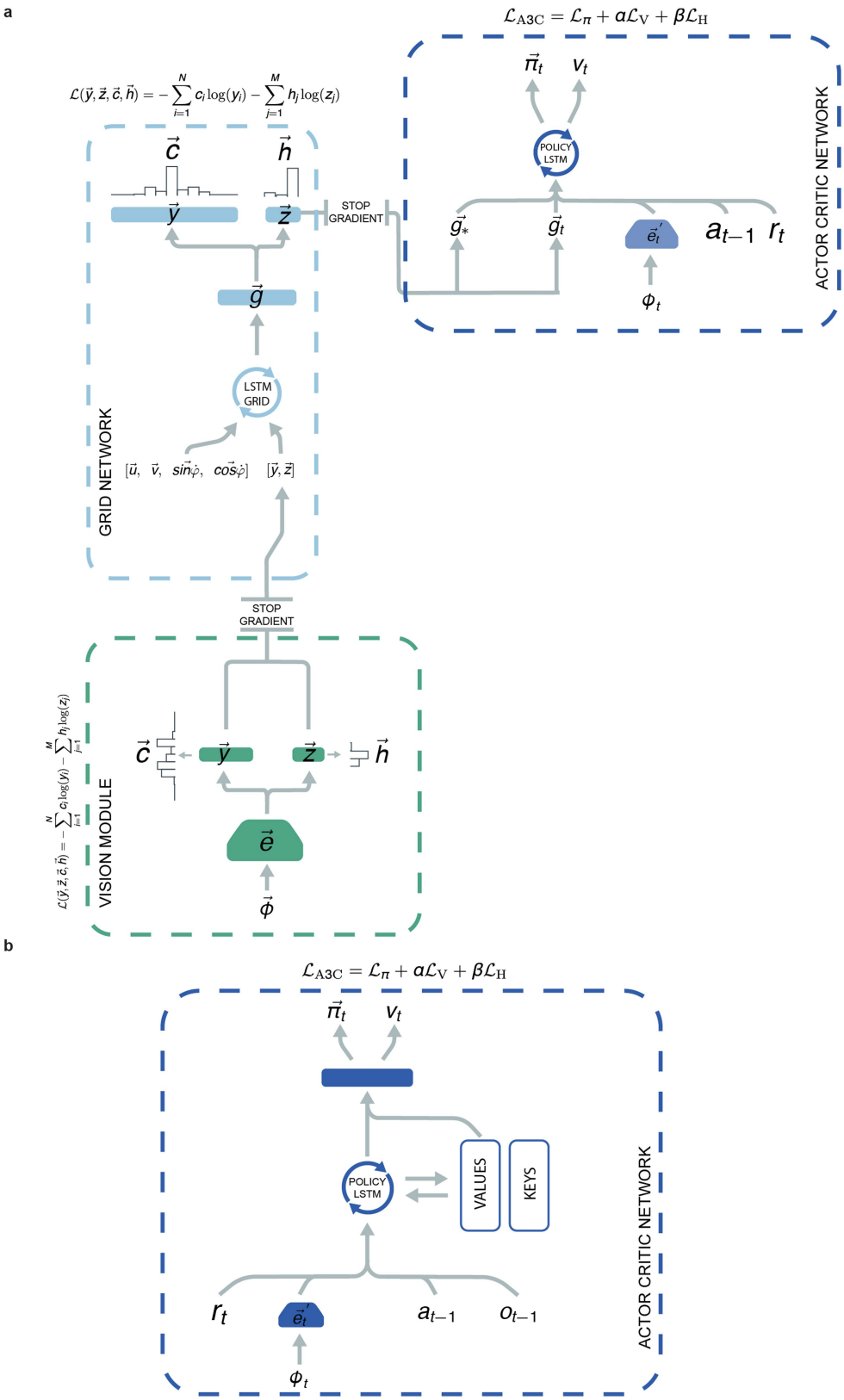
Extended Data Fig. 6 | Characterization of grid-like representations and robustness of performance for the grid cell agent in the square land maze environment. a, Spatial activity plots for the 256 linear layer units in the agent exhibit spatial patterns similar to grid, border, and place cells. **b**, Cumulative reward indexing goal visits per episode (goal, 10 points) when distal cues are removed (dark blue) and when distal cues are present (light blue). Performance is unaffected, hence dark blue largely obscures light blue. Average of 50% best agent replicas ($n = 32$) plotted (see Methods). The grey band displays the 68% CI based on 5,000 bootstrapped samples. **c**, Cumulative reward per episode when no goal code was provided (light blue) and when goal code was provided (dark blue). When no goal code was provided the agent performance fell to that of the baseline deep reinforcement learning agent (A3C) (100 episodes average score no goal code, 123.22 versus A3C, 112.06; effect size, 0.21; 95% CI, 0.18–0.28). Average of 50% best agent replicas ($n = 32$) plotted (see Methods). The grey band displays the 68% CI based on 5,000 bootstrapped samples. **d**, After locating the goal for the first time during an episode, the agent typically returned directly to it from each new starting position, showing decreased latencies for subsequent visits, paralleling the behaviour exhibited by rodents.



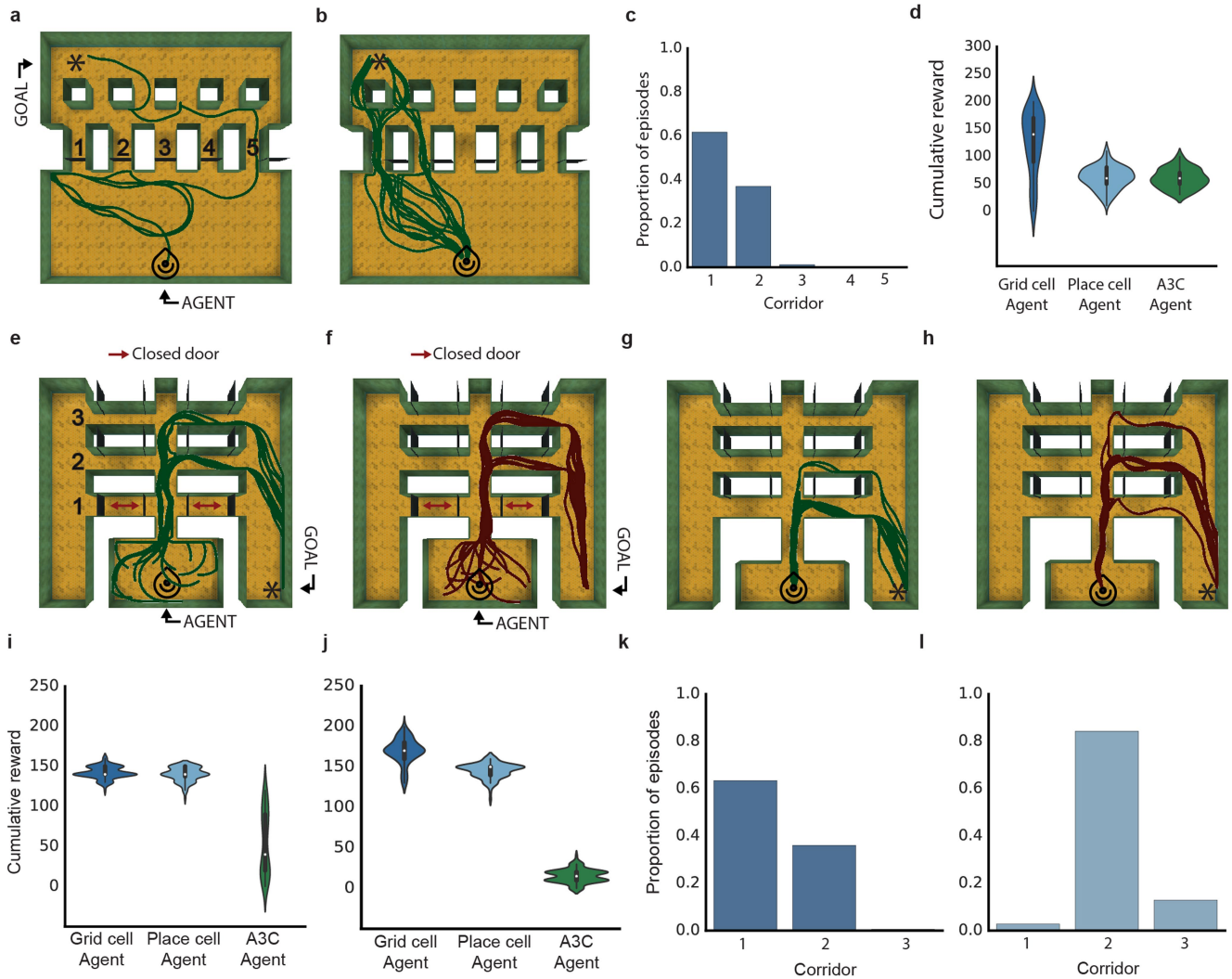
Extended Data Fig. 7 | Robustness of grid cell agent and performance of other agents. a–c, AUC performance gives robustness to hyperparameters (that is, learning rate, baseline cost, entropy cost; see Supplementary Table 2 in Supplementary Methods for details of the range) and seeds (see Methods). For each environment we run 60 agent replicas (see Methods). Light purple is the grid agent, blue is the place cell agent and dark purple is A3C. **a**, Square arena. **b**, Goal-driven. **c**, Goal doors. In all cases the grid cell agent shows higher robustness to variations in hyperparameters and seeds. **d–i**, Performance of place cell prediction, NavMemNet and DNC agents (see Methods) against grid cell agent. Dark blue is the grid cell agent (Extended Data Fig. 5), green is the place cell prediction agent (Extended Data Fig. 9a), purple is the DNC agent, light blue is the NavMemNet agent (Extended Data Fig. 9b). The grey band displays the 68% CI based on 5,000 bootstrapped samples. **d–f**, Performance in goal-driven. **g–i**, Performance in goal-doors. Note that the performance of the place cell agent (Extended Data Fig. 8b, lower panel) is shown in Fig. 3.



Extended Data Fig. 8 | Architecture of the A3C and place cell agent. **a**, The A3C implementation is as described[41]. **b**, The place cell agent was provided with the ground-truth place, $\vec{c}_t$, and head-direction, $\vec{h}_t$, cell activations (as described above) at each time step. The output of the fully connected layer of the convolutional network $\vec{e}_t$ was concatenated with the reward r_t, the previous action a_{t-1}, the ground-truth current place code, $\vec{c}_t$, and current head-direction code, $\vec{h}_t$, together with the ground truth goal place code, $\vec{c}_*$, and ground truth head direction code, $\vec{h}_*$, observed the last time the agent reached the goal (see Methods).



Extended Data Fig. 9 | Architecture of the place cell prediction agent and of the NavMemNet agent. a, The architecture of the place cell prediction agent is similar to the grid cell agent, having a grid cell network with the same parameters as that of the grid cell agent. The key difference is the nature of the input provided to the policy LSTM. Instead of using grid codes from the linear layer of the grid network $\vec{g}$, we used the predicted place cell population activity vector $\vec{y}$, and the predicted head direction population activity vector $\vec{z}$, (the activations present on the output place and head direction unit layers of the grid cell network, corresponding to the current and goal position, respectively) as input for the policy LSTM. As in the grid cell agent, the output of the fully connected layer of the convolutional network, $\vec{e}$, the reward r_t, and the previous action a_{t-1}, were also input to the policy LSTM. The convolutional network had the same architecture as described for the grid cell agent. **b**, NavMemNet agent. The architecture implemented is as described[3], specifically FRMQN, but the A3C algorithm was used in place of Q-learning. The convolutional network had the same architecture described for the grid cell agent and the memory was formed of two banks (keys and values), each composed of 1,350 slots.



Extended Data Fig. 10 | Flexible use of shortcuts. a, Overhead view of the linear sunburst maze in initial configuration, with only door 5 open. Example trajectory from grid cell agent during training (green line, icon indicates start location). **b**, Test configuration with all doors open; grid cell agent uses the newly available shortcuts (multiple episodes shown). **c**, Histogram showing proportion of times the agent uses each of the doors during 100 test episodes. The agent shows a clear preference for the shortest paths. **d**, Performance of grid cell agent and comparison agents during test episodes. **e**, **f**, Example grid cell agent (**e**) and example place cell agent (**f**) trajectory during training in the double E-maze (corridor 1 doors closed). **g**, **h**, In the test phase, with all doors open, the grid cell agent exploits the available shortcut (**g**), while the place cell agent does not (**h**). **i**, **j**, Performance of agents during training (**i**) and test (**j**). **k**, **l**, The proportion of times the grid (**k**) and place (**l**) cell agents used the doors on the first to third corridors during test. The grid cell agent shows a clear preference for available shortcuts, while the place cell agent does not.

Corresponding author(s): abanino@google.com, caswell.barry@gmail.com, dkumaran@google.com

Life Sciences Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form is intended for publication with all accepted life science papers and provides structure for consistency and transparency in reporting. Every life science submission will use this form; some list items might not apply to an individual manuscript, but all fields must be completed for clarity.

For further information on the points included in this form, see Reporting Life Sciences Research. For further information on Nature Research policies, including our data availability policy, see Authors & Referees and the Editorial Policy Checklist.

Please do not complete any field with "not applicable" or n/a. Refer to the help text for what text to use if an item is not relevant to your study.
For final submission: please carefully check your responses for accuracy; you will not be able to make changes later.

Experimental design

1. Sample size

Describe how sample size was determined.

Score comparisons for the agent:
For this analysis no a priori power calculations were conducted, as there is no substantive prior evidence to predict the effect sizes. So we ran 100 testing episodes, to ensure sufficient variability in the configuration of the environment -- in line with current practice for agent evaluation. Effect size are reported.

Metric code decoding analysis:
For this analysis no a priori power calculations were conducted, as there is no substantive prior evidence to suggest the effected effect sizes. However, we note that for a correlation analysis, a sample size of 100 would be sufficient for an expected Pearson coefficient of 0.3 or above (given alpha of 0.05, and beta of 0.2).

Assessment of grid regularity (gridness) and grid scale:
For this analysis no a priori power calculations were conducted. All units in the linear dropout layer of the 'grid cell network' were assessed to determine their gridness. Only units with gridness greater than the 99th percentile of a shuffled distribution (described in methods) were deemed to be 'grid-like' - following the procedure described in Barry et al, 2017 and other standard procedure (e.g. Yartsev et al., 2011)

2. Data exclusions

Describe any data exclusions.

Score comparisons for the agent:
no data were excluded

Metric code decoding analysis:
No data were excluded.

Assessment of grid regularity (gridness) and grid scale:
All linear layer units were assessed for gridness - none were excluded. Assessment of grid scale was limited to units determined to be grid-like.

3. Replication

Describe the measures taken to verify the reproducibility of the experimental findings.

Supervised Learning experiments demonstrating grid-like representations:
We ran 100 experiments using different network initializations showing the effect to be robust

Deep reinforcement learning experiments: Agent Performance
For all agents we ran many replicas and seeds in all environments to show that performance was robust.

4. Randomization

Describe how samples/organisms/participants were allocated into experimental groups.

Score comparisons for the agent:
Given the specific analysis there was no need to allocate the samples into different group

Metric code decoding analysis:
We compared three types of artificial agent. For each agent we were interested in mapping various metrics to the internal state on different episodes. There was therefore no allocation of samples into discrete groups.

Assessment of grid regularity (gridness) and grid scale:
Given the specific analysis there was no need to allocate the samples into different group.

5. Blinding

Describe whether the investigators were blinded to group allocation during data collection and/or analysis.

Score comparisons for the agent:
No group allocation was involved.

Metric code decoding analysis:
No group allocation was involved.

Assessment of grid regularity (gridness) and grid scale:
No group allocation was involved.

Note: all in vivo studies must report how sample size was determined and whether blinding and randomization were used.

6. Statistical parameters

For all figures and tables that use statistical methods, confirm that the following items are present in relevant figure legends (or in the Methods section if additional space is needed).

n/a	Confirmed
☐	☒ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement (animals, litters, cultures, etc.)
☐	☒ A description of how samples were collected, noting whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒ A statement indicating how many times each experiment was replicated
☐	☒ The statistical test(s) used and whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of any assumptions or corrections, such as an adjustment for multiple comparisons
☐	☒ Test values indicating whether an effect is present *Provide confidence intervals or give results of significance tests (e.g. P values) as exact values whenever appropriate and with effect sizes noted.*
☐	☒ A clear description of statistics including central tendency (e.g. median, mean) and variation (e.g. standard deviation, interquartile range)
☐	☒ Clearly defined error bars in all relevant figure captions (with explicit mention of central tendency and variation)

See the web collection on statistics for biologists for further resources and guidance.

Software

Policy information about availability of computer code

7. Software

Describe the software used to analyze the data in this study.

Metric code decoding analyses were implemented using the Python toolbox Sklearn. Assessment of gridness and grid scale was conducted with bespoke code written in Matlab 2016b. Agent comparison were conducted using Python

For manuscripts utilizing custom algorithms or software that are central to the paper but not yet described in the published literature, software must be made available to editors and reviewers upon request. We strongly encourage code deposition in a community repository (e.g. GitHub). *Nature Methods* guidance for providing algorithms and software for publication provides further information on this topic.

Materials and reagents

Policy information about availability of materials

8. Materials availability

Indicate whether there are restrictions on availability of unique materials or if these materials are only available for distribution by a third party.

No unique materials were used in the study.

9. Antibodies

Describe the antibodies used and how they were validated for use in the system under study (i.e. assay and species).

No antibodies were used in the study.

10. Eukaryotic cell lines

a. State the source of each eukaryotic cell line used. — No eukaryotic cell lines were used in the study.

b. Describe the method of cell line authentication used. — No eukaryotic cell lines were used in the study.

c. Report whether the cell lines were tested for mycoplasma contamination. — No eukaryotic cell lines were used in the study.

d. If any of the cell lines used are listed in the database of commonly misidentified cell lines maintained by ICLAC, provide a scientific rationale for their use. — No eukaryotic cell lines were used in the study.

Animals and human research participants

Policy information about studies involving animals; when reporting animal research, follow the ARRIVE guidelines

11. Description of research animals

Provide all relevant details on animals and/or animal-derived materials used in the study. — No animals were used in the study.

Policy information about studies involving human research participants

12. Description of human research participants

Describe the covariate-relevant population characteristics of the human research participants. — The study did not involve human research participants