No Free Lunch from Deep Learning in Neuroscience: A Case Study through Models of the Entorhinal-Hippocampal Circuit

Rylan Schaeffer
Computer Science
Stanford
rschaef@cs.stanford.edu

Mikail Khona
Physics
MIT
mikail@mit.edu

Ila Rani Fiete
Brain and Cognitive Sciences
McGovern Institute for Brain Research
MIT
fiete@mit.edu

Abstract

Research in Neuroscience, as in many scientific disciplines, is undergoing a renaissance based on deep learning. Unique to Neuroscience, deep learning models can be used not only as a tool but interpreted as models of the brain. The central claims of recent deep learning-based models of brain circuits are that they make novel predictions about neural phenomena or shed light on the fundamental functions being optimized. We show, through the case-study of grid cells in the entorhinal-hippocampal circuit, that one may get neither. We begin by reviewing the principles of grid cell mechanism and function obtained from first-principles modeling efforts, then rigorously examine the claims of deep learning models of grid cells. Using large-scale hyperparameter sweeps and theory-driven experimentation, we demonstrate that the results of such models may be more strongly driven by particular, non-fundamental, and post-hoc implementation choices than fundamental truths about neural circuits or the loss function(s) they might optimize. We discuss why these models cannot be expected to produce accurate models of the brain without the addition of substantial amounts of inductive bias, an informal No Free Lunch result for Neuroscience. Based on first principles work, we provide hypotheses for what additional loss functions will produce grid cells more robustly. In conclusion, caution and consideration, together with biological knowledge, are warranted in building and interpreting deep learning models in Neuroscience.

1 Introduction

Over the past decade, deep learning (DL) has underpinned nearly every success story in machine learning, e.g., [49, 5] and increasingly many advances in fundamental science research, e.g., [29]. In neuroscience, DL is similarly gaining widespread adoption as an indispensable method for behavioral and neural data analysis [45, 43, 23, 37, 34, 40].

But DL offers a unique contribution to neuroscience that goes beyond its role in other fields, in that deep networks can be viewed as models of the brain. The success of DL in matching or surpassing human performance suggests it is now possible to construct models of circuits that may underlie human intelligence. As a recent review wrote, "researchers are excited by the possibility that deep

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

neural networks may offer theories of perception, cognition and action for biological brains. This approach has the potential to radically reshape our approach to understanding neural systems" [46]. Further, DL is a democratizing force for building neural circuit models of complex function.

The essential claims (and promises) of DL-based models of the brain are that 1) Because the models are trained on a specific optimization problem, if the resulting representations match what has been observed in the brain, then the models reveal which optimization problem(s) the brain is solving, or 2) These models, when trained on sensible optimization problems, should generate novel predictions about the brain's representations and behavior.

However, given the nascent nature of such approaches and the excitement accompanying some claims, we should examine them carefully. In DL, some successes attributed to novel algorithms have been shown to instead stem from seemingly minor or unstated implementation choices [57, 17, 28]. In this paper, we ask whether Neuroscientists should similarly be cautious that DL-based models of neural circuits that make specific claims about revealing the brain's optimization functions or that predict specific neural tuning curves may tell us less about fundamental scientific truths and more about programmers' particular implementation choices, and might as a result be more post hoc than predictive.

To explore these questions and point toward a systematic and circumspect use of DL for Neuroscience, we evaluate recent DL-based models of grid cells in the entorhinal-hippocampal circuit. The medial entorhinal cortex (MEC) and hippocampus (HPC) are part of the hippocampal formation, a brain structure critical for learning and memory. In a pair of Nobel-prize winning discoveries, HPC was shown to contain **place cells** [42], and MEC, its cortical input, was shown to contain **grid cells** [25]. Place cells fire at one or several seemingly random locations in small and large environments [44], while grid cells fire in a spatially periodic pattern in all two-dimensional environments, whenever the animal is at the vertices of a hexagonal lattice [25]. Over five decades, the hippocampal formation has been central to understanding how the brain organizes spatial and episodic memory, for experimentalists and theorists alike, with many mysteries remaining. A recent series of DL-based models of the circuit [13, 3, 51, 60, 41]) present a story that training neural circuits on **path integration (PI)** (i.e., the task of estimating one's spatial position in an environment by integrating velocity estimates) results in the emergence of grid cells.

We use public code from prior publications to demonstrate these results are due to implementation details that tell us more about those choices than they do about MEC. By leveraging theoretically-guided large-scale hyperparameter exploration and hypothesis-driven experimentation, we show:

1. Networks trained on path integration tasks almost always learn to optimally encode position, but almost never learn grid-like representations to do so.

2. The emergence of grid-like representations depends wholly on a specifically chosen encoding of the supervised target, not on the task itself, and these choices may be unrealistic.

3. The chosen encoding requires many other sensitive hyperparameter choices, such that small alterations result in loss of grid-like representations.

4. Grid periods and period ratios depend on hyperparameter choices and are not set by the task.

5. Multiple grid modules, a fundamental characteristic of the grid cell system, do not emerge.

In short, deep learning models of MEC-HPC yield grid-like units only when many specific and biologically implausible implementation choices are made, intentionally baking grid-like units into the task-trained networks, and the emergent grid-like units lack key grid cell properties. It is highly improbable that DL models of path integration would have produced grid cells as a novel prediction from task-training, had grid cells not already been known to exist. Moreover, it is unclear what interpretability or understanding these models contribute, beyond or even up to what has been shown for these circuits by existing models. These results challenge the notion that deep networks offer a free lunch for Neuroscience in terms of discovering the brain's optimization problems or generating novel a priori predictions about single-neuron representations, and warn that caution is needed when building and interpreting such models.

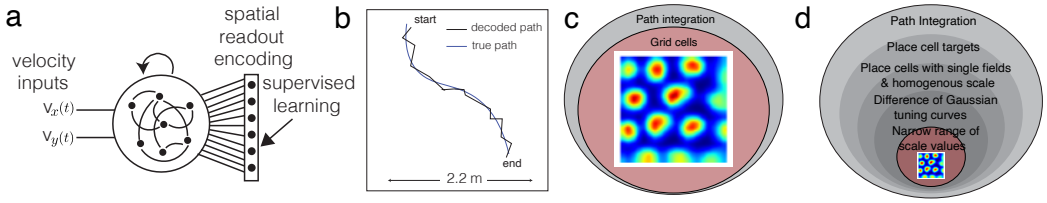

Figure 1: (a-b) Schematic of approach of training recurrent networks to predict (different possible encodings of) 2D position from 2D velocity, in a supervised manner. (b-c) Grid cells are shown to emerge in recent DL papers as a consequence of optimizing a path integration objective [13, 3, 51, 60, 41], with the suggestion that path integration implies grid cells. (d) In the current work, we show how most artificial neural networks (ANNs) trained to path integrate can do so, but only a very small set of output encoding functions (vanishingly small in the function space) and a very small fraction of hyperparameter space yield grid cells, leading to the conclusion that grid cell emergence results in ANNs are post hoc: they result from setting hyperparameters to bake grid-like units into networks.

Our work benefited greatly from previous publications that published their code, for which the authors should be commended. By building on their code, we have been able to present novel insights that we hope will contribute to a clearer understanding of the potential risks and rewards of using and interpreting DL models in Neuroscience. To facilitate further research, we similarly release code[1].

2 Background: Grid Cells

Grid cells [25] are found in the medial entorhinal cortex of mammals and are tuned to represent the spatial location of the animal as it traverses 2D space. Each cell fires at every vertex of a triangular lattice that tiles the explored space, regardless of the speed and direction of movement through the space. As a population, grid cells exhibit several striking properties that provide support for a specialized and modular circuit. Grid cells form discrete modules (clusters), such that all cells within a module share a common period and orientation, while different modules express discretely different spatial periods [54]. The period ratios of successive modules have values in the range of 1.2-1.5.

The mechanism underlying grid cells is widely supported to be through attractor dynamics: Translation-invariant lateral connectivity within the grid cell network results in a linear Turing instability and pattern formation [8, 19, 6]. These models explain how grid cells can convert velocity inputs into updated spatial estimates, and make several predictions that have been confirmed in experiments, including most centrally the stability of low-dimensional cell-cell relationships regardless of environment and behavioral state, that define a toroidal attractor dynamics [20, 61, 56, 22, 21].

3 Experimental approach

Path integration (PI) is the task of using self-velocity estimates to track one's spatial position, a crucial component of spatial navigation. The central message of existing DL models of grid cells is that training ANNs to path integrate causes the networks to learn grid cells [13, 3, 51, 60, 41].

We follow the setup used by many previous papers: a 2.2 m x 2.2 m arena is created, then, spatial trajectories (i.e. sequences of positions and velocities) are sampled. Networks receive as inputs the initial position and velocities, and are trained to output (a possible encoding of) the positions in a supervised manner (Fig. 1ab). There are multiple possible encodings of position, and, as we will show, this choice is critical. Two simple encodings are Cartesian [13] or polar [1]. Another encoding scheme is via bump functions in 2D space, with each output assigned a single different position that collectively tiles the space and all outputs assigned identical tuning curves [3, 51, 51, 41]. This encoding has been equated with place cells, even though place cells' fields tend to be heterogeneous in size and shape [44, 15], as well as in number [44], and, unlike the choice of a difference-of-Gaussians (DoG) or difference-of-Softmax (DoS) readout tuning in ANN models, do not exhibit any inhibitory surround. See Appendix A for position encoding details. For all encodings, supervised learning is used to train the network via backpropagation through time.

[1] `https://github.com/FieteLab/NeurIPS-2022-No-Free-Lunch`

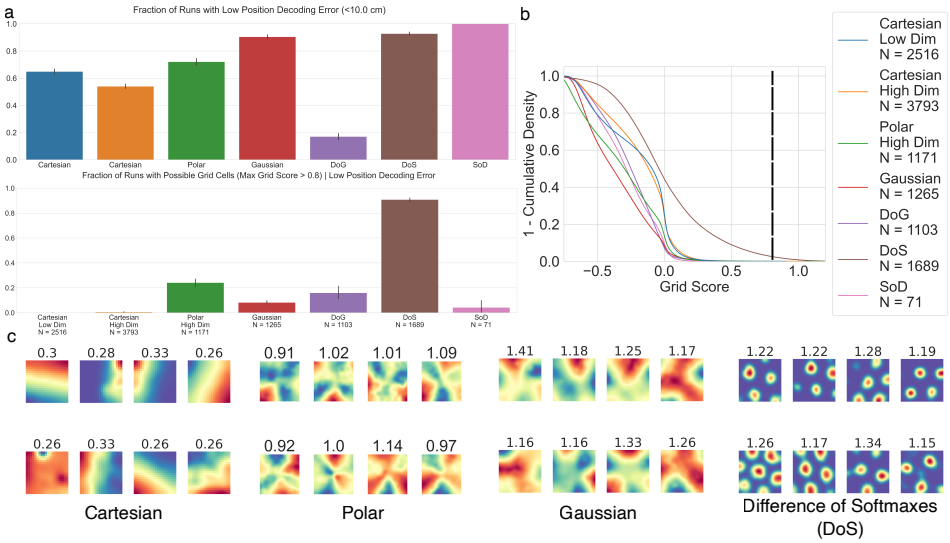

Figure 2: (a) Top: Across readout encodings, almost all networks learn to optimally encode position. Bottom: Few networks display possible grid-like representations (grid score threshold = 0.8). (b) Kernel density estimates of grid scores per readout encoding. (c) Rate maps of high grid-scoring units in deep networks trained on i) Cartesian, ii) Polar, iii) Gaussian, iv) specifically selected (tuned) Difference-of-Softmaxes (DoS) readouts. i)-iii) do not learn any grid cells. Only networks trained on DoS readouts display grid-like cells. Numbers above rate maps are grid scores.

Spatial tuning assessments The spatial tuning of hidden units in the networks, visualized as ratemaps, is the primary method for comparison with the brain's grid cells. To compute ratemaps, a trained network is evaluated on long trajectories that cover the 2D environment, then each hidden unit's activity is binned against the true position in the environment, and counts per bin are normalized by the number of visits to that bin. Ratemaps of units are compared with grid cells through the gridness score. **We are extremely lenient with classifying a training run with particular hyperparameters a success: if even a *single* hidden unit has a grid score above a certain threshold, we say the model possibly possesses grid cells.** The grid score is not perfect since cells classified by grid scores represent only an upper bound on the total number of grid cells. For details, see Appendices B and C.

4 Networks trained on path integration tasks learn to estimate position, but rarely learn grid cells

As [51] wrote in their section titled "Optimally encoding position yields diverse grid-like patterns," "Why do these diverse architectures, across diverse tasks (both navigation and autoencoding), all converge to a grid-like solution, and what governs the lattice structure of this solution?" In contrast, we demonstrate that most networks do not converge to a grid-like solution, instead requiring very specific readout tuning functions and hyperparameter choices. Grid-like representation emergence is determined by choices made by the programmer that, rather than relating to the path integration objective or biologically realistic place cells, is designed post hoc to produce grid cells.

We ran large-scale hyperparameter sweeps across common implementation choices: 1) Architectures: RNN [16]; LSTM [27]; GRU [11]; UGRNN [12]; 2) Activation: Sigmoid; Tanh; ReLU; Linear; 3) Optimizers: SGD, Adam [35]; RMSProp [26] 4) Supervised Targets: Cartesian; Polar; high-dimensional bump-like readout population code with Gaussian [3], **Difference-of-Softmaxes (DoS)** [51, 52, 41] or Difference-of-Gaussians (DoG) tuning curves. 5) Loss: mean squared error on the agent's Cartesian position [30, 13]; geodesic distance on the agent's polar position [1]; cross entropy on a high-dimensional population of bump-like readout units [3, 51, 41] 6) Miscellaneous: recurrent & readout dropout, initialization, regularization, seed.

For networks with bump-like population readouts, we additionally swept: 1) Field width σ, i.e., standard deviation of the Gaussian tuning curve (often denoted $\sigma_E \stackrel{\text{def}}{=} \sigma$ in the literature); 2) Whether the bump-like readouts have homogeneous or heterogeneous field widths; 3) In the case of readouts with

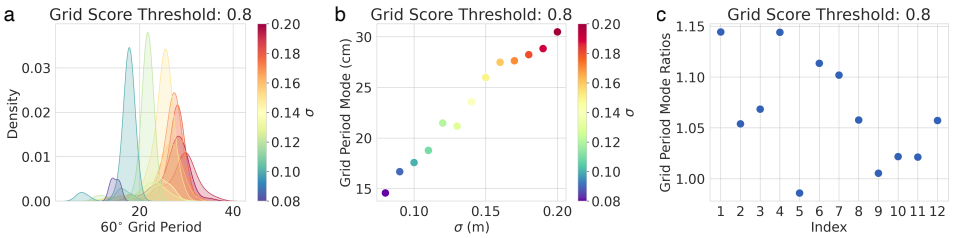

Figure 3: **Grid periods are hyperparameter-dependent and multiple modules do not emerge.** (a) Over a wide sweep of DoS (the most likely to produce grid cells, Fig. 2a bottom) target field widths σ_E, the distribution of grid periods is unimodal (each color is distribution of periods obtained from 3 runs with a σ_E value; different periods are only obtained by varying σ_E), meaning multiple grid modules do not emerge, in contrast to real grid cells. (b) The chosen target field width σ_E determines the grid period mode, meaning that hand-designed hyperparameter choices, not an intrinsic emergent property, sets the grid period. (c) If we use grid periods obtained from smoothly sweeping σ_E as a proxy for different modules, the period ratios are closer to 1 than the experimental ratios of ~ 1.4.

DoG or DoS tuning curves, the surround scale s, i.e., the ratio between the inhibitory and excitatory Gaussian's standard deviations ($s \stackrel{\text{def}}{=} \sigma_I/\sigma_E$); 4) Number of fields per readout unit. Evaluating the entire hyperparameter volume is computationally prohibitive, so we evaluated a subvolume most consistent with previous papers, focusing on perturbations from hyperparameters that did produce grid cells. In this sense, **our search was biased toward configurations shown to produce grid cell emergence and thus our findings about the fragility of these solutions conservatively favored these solutions as much as possible.** All sweeps are provided in Appendix E.

To evaluate whether a network learns to optimally encode its position, we measured its position decoding error using previous papers' method [3, 51, 41]: we computed position decoding error using the network's output Cartesian positions (if trained on Cartesian targets) or by decoding position from the network's outputs. Any network with error < 10 cm was considered to have achieved optimal position encoding; this threshold was chosen based on noise inherent in the decoding algorithm.

In total, we trained $> 11,000$ networks and found that most hyperparameter configurations succeed in learning to path integrate (Fig. 2a, Top), but few learn grid cell representations (Fig. 2a, Bottom). This is consistent with earlier work [30, 1] demonstrating that networks can learn to path integrate and solve other navigational problems (e.g. estimating which of several environments correspond to the current location) without grid-like units emerging as a solution.

5 Grid-like unit emergence requires a highly specific choice of supervised target encoding

We next sought to characterize when grid cells are learnt under different encodings of the supervised target, i.e., 2D position. We tested multiple encodings: i) Cartesian, ii) Polar, iii) Gaussian, iv) a very specifically selected Difference-of-Softmaxes (DoS) readout and more. We found that grid cells do not emerge from Cartesian, Polar or Gaussian encodings (Fig. 2abc), consistent with earlier work [30]. Only by choosing a DoS readout tuning curve did grid cells sometimes emerge.

5.1 Grid periods are parameter-dependent, and multiple modules do not emerge

Next, a prominent feature of grid cells that is critical for their unambiguous encoding of position over large scales is the existence of a discrete set of grid periodicities, which tend to scale by a rough factor of 1.4 between adjacent scales [54]. We asked whether ANN models generate multiple periods and when they did so, what hyperparameters and other choices the formed periods depended on.

To ensure we would obtain at least some grid cells, we fixed the readouts to be DoSs, and swept over different scales of the DoS. We found that almost all runs had a unimodal distribution of grid periods (Fig. 3a), meaning the networks learnt only one module of grid cells. During the NeurIPS review process, other researchers independently found the same result [48].

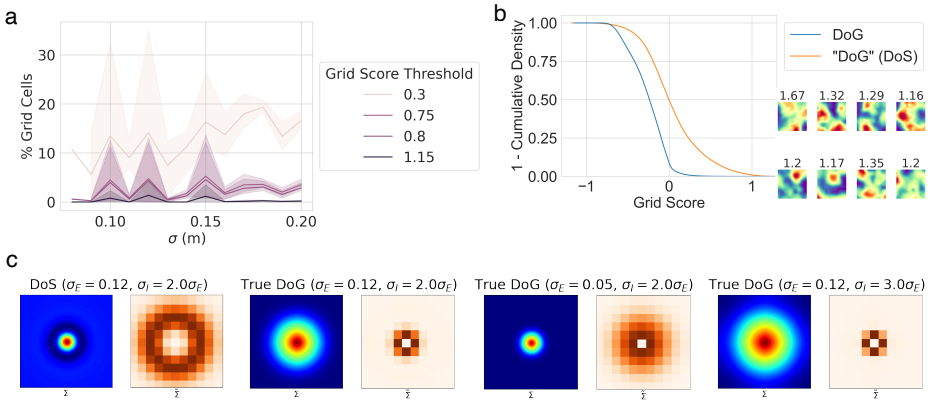

Figure 4: **Other changes to Difference-of-Softmaxes (DoS) affect the formation of grid cells:** (a) The existence of grid solutions even with DoS readout encodings is highly sensitive to parameters such as the target function receptive field width σ_E, and small alterations result in the disappearance of grid cells regardless of grid score threshold. (b) (left) Comparison of grid scores of trained networks with DoG shows that the DoS, as used by [51, 41, 52], is critical for high grid scores (right). Rate maps of highest scoring units from DoG networks achieving low position decoding error. (c) Computing the Fourier transform (right of each pair) of the readout second-moment matrix (left of each pair) explains that DoS places Fourier power on an annulus with sufficiently large radius to produce grid cells, showing why the particular DoS choice is critical.

Further, we found that the period of the formed grid-like representation is completely determined by the scale σ_E of the externally imposed readout DoS (Fig. 3). The period of the grid-like responses in every run increased monotonically with the width of the DoS readout (Fig. 3b). Since the models did not result in multiple modules in a single network, we used the somewhat discrete distribution of peaks of the single module formed when sweeping the DoS parameter more continuously to compute grid period ratios. These period ratios from adjacent peaks led to non-biological values (Fig. 3c).

5.2 Fourier analysis of Turing instability explains the preceding empirical results

Why do only Difference-of-Softmaxes (DoS) readouts produce grid-like units? We restate the essence of previous analyses of first-principles models [8, 32] here, and explain why they might explain our findings.

In a recurrent network with dynamics $\dot{r}(x) = -r(x) + g(W \star r)$, where x designates the neural index (in a continuum approximation for neurons), $W \star r$ designates the total (integrated) inputs from the network to the neuron at index x, and g is the non-linearity, if the recurrent weight interaction is translationally invariant, then $W(x, x') = W(x - x') = W(\Delta x)$. Under DoG interactions:

$$W(\Delta x) \equiv f(\Delta x) = \alpha_E \exp\Big(-\frac{(\Delta x)^2}{2\sigma_E^2}\Big) - \alpha_I \exp\Big(-\frac{(\Delta x)^2}{2\sigma_I}\Big) \tag{1}$$

where Δx refers to the difference of indices between the neural pair linked by the weights. The evolution of activity can be decomposed into the growth and decay of Fourier components of the rate vector, which is fully determined by the Fourier transform of W, which is given by:

$$\tilde{f}(k) = \int_{\mathcal{R}} d(\Delta x) f(\Delta x) e^{ik\Delta x} = \alpha_E \sigma_E \exp\left(-\frac{\sigma_E^2 k^2}{2}\right) - \alpha_I \sigma_I \exp\left(-\frac{\sigma_I^2 k^2}{2}\right) \tag{2}$$

Here α_E (α_I) denotes the strength and σ_E (σ_I) denotes the scale of excitation (inhibition). For linearized dynamics that approximate $\dot{r}(x) \sim -r(x) + f(\Delta x) \star r$ (i.e., g has been linearized), the solution will be periodic if the maxima of $\tilde{f}(k)$, given by $[k^*]^2 = \frac{2}{\sigma_E^2 - \sigma_I^2} \log\left(\alpha_E \sigma_E^3 / \alpha_I \sigma_I^3\right)$, contains sufficient power and if $k^* \neq 0$. Specifically, the condition for pattern formation is $\tilde{f}(k^*) > 1$ [9, 32]. In particular, the inhibitory surround contained in $f(\Delta x)$, with strength σ_I, is key to pattern

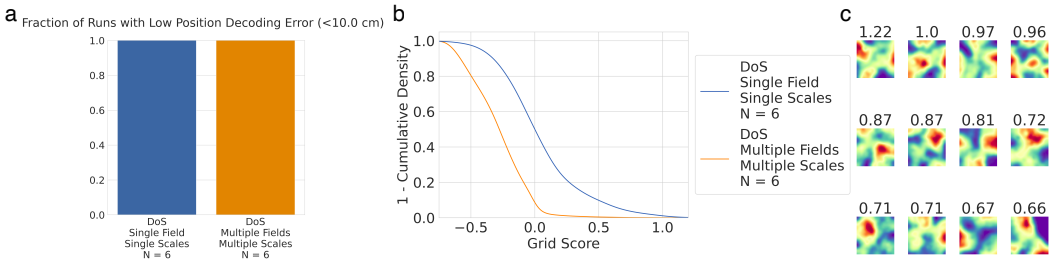

Figure 5: **Adding place cell-like heterogeneity to the readouts prevents the formation of grid cells.** To maximally favor the emergence of grid representations, we selected DoS encodings with the best hyperparameters for grid cell emergence (RNN or UGRNN, ReLU, $\sigma_E = 0.12$ cm, $s = 2.0$, 3 seeds), then tested the effect of multiple fields per place cell ($\sim 1 + \text{Pois}(3.0)$) and multiple scales (receptive field width $\sigma_E \sim \text{Unif}(0.06, 1.0)$ m and surround scale $s \sim \text{Unif}(1.25, 4.5)$). (a) Networks trained on single-field single-scale DoS and multi-scale multi-field DoS (more similar to biological place cells) readouts all obtain low position decoding error. (b) Multi-scale multi-field DoS readouts do not learn grid cells. (c) Highest-scoring rate maps from multi-field multi-scale networks.

formation; if $\sigma_I \to \infty$ or $\alpha_I \to 0$, the maximum is at the origin ($k^* = 0$), causing no pattern formation. Therefore, a Gaussian interaction cannot produce periodic patterns.

The connection of this theory to ANNs trained with supervised readout target P was made in [51] through the observation that dynamics with an MSE loss $||P - W_{\text{readout}} r||^2$ can be approximated as $\dot{r} = -\lambda r + \Sigma r$, where $\Sigma = PP^T$ is the readout correlation matrix and λ is a regularization parameter. This dynamics now looks identical to the first dynamics with recurrent interaction matrix $W(x, x') = \Sigma_{xx'}$. For identical, unimodal target readout functions with DoG tuning curves, this readout correlation matrix is also a difference of Gaussians (Appendix D). By the same theory outlined above, Gaussian tuning curves should not produce periodic patterns.

5.3 Unmentioned implementation detail matters

We discovered a seemingly minor implementation detail in several preceding papers [51, 52, 41] that proved necessary for the emergence of grid cells, that (to the best of our knowledge) is not mentioned in the main texts nor supplements. While these papers nominally refer to a Difference-of-Gaussian (DoG) readout target function (Eqn. 1), their code instead uses a Difference-of-Softmaxes (DoS) target function. When we trained ideal grid-forming ReLU networks on place-cell-like target encodings with DoG tuning curves, sweeping the receptive field σ and surround scale s, DoG readouts did not result in grid cells (Fig. 4b), while DoS readouts did. The preceding Fourier analysis shows why DoS tuning, though still more unrealistic as a place cell model than DoG, matters (Fig. 4c).

6 Grid cells disappear with more biologically-realistic readout functions

Place cells, to which grid cells project, differ significantly from the highly simplified single-scale, single-field ANN readouts. Place cells have heterogeneously distributed field widths, many with multiple fields [44, 15] (and are not DoS-like functions). This naturally leads to the question: Will readouts targets with multiple scales and multiple fields per place cell still produce grid cells? We found that networks trained with multiple-field multiple-scale bump encodings per readout unit achieve comparably low position decoding error as good single-field single-scale encodings (Fig. 5a), but do not learn grid cells (Fig. 5bc). This further demonstrates that accurate position encoding and integration do not require grid representations.

7 Why path-integrating ANNs might achieve high predictivity of MEC data

We conclude by introducing a puzzle. A recent NeurIPS spotlight [41] notes that networks trained on single-field single-scale DoS readout encodings explain variance in mouse MEC neural activity at nearly 100% of variance explained by other mice. In contrast, our results demonstrate that these networks learn few grid cells and produce unimodal grid period distributions inconsistent with

biological grid cells, and require readout encodings inconsistent with biological place cells to do so. How are these networks able to predict mouse MEC neural activity so well?

The neural data is not public and the analysis code is not open source, so we are unable to investigate this puzzle. However, we offer a conjecture with preliminary evidence. The analysis of [41] linearly regressed rate maps from one agent (mouse or network) onto rate maps from another mouse, and used Pearson correlation as a measure of "neural predictivity." We conjecture that different architecture-activation pairs achieve different neural predictivity scores because different pairs learn different intrinsic dimensionalities that then provide richer/ poorer bases for linear regressions.

To explore our conjecture, we trained [41]'s 5 architectures (RNN, LSTM, GRU, UGRNN, SRNN) with ReLU activations, with 5 seeds per architecture-activation pair. For each trained network, we extracted rate maps and computed a standard linear measure of dimensionality called participation ratio [36], then plotted each pair's average participation ratio against the published neural predictivity. We found networks with higher dimensional rate maps have higher neural predictivity (Fig. 6).

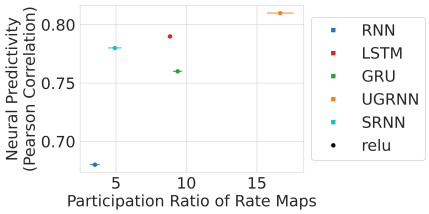

Figure 6: **Networks with higher (lower) dimensional rate maps display higher (lower) "neural predictivity" of mouse MEC rate maps.** Each dot is an architecture-activation pair; participation ratio averaged over 5 seeds. Neural predictivity from [41].

We caution this correlation between dimensionality and neural predictivity is not strong evidence. However, we predicted that similar results would be found in other species and modalities (e.g., vision, audition), and during the NeurIPS review process, two independent research groups confirmed our predictions in macaque vision [2] and human audition [58]. If our conjecture is correct, it raises an important research question of whether linear regression-based comparisons of ANNs with neural data might produce better matches to biology more because ANNs have higher dimensional representations than competitors, rather than because of a detailed similarity. We hope this section will spur further analysis of methods to compare artificial and biological networks.

8 Discussion

For research that uses deep networks as models of the brain, there is a fundamental obstacle to making the claim that a given optimization problem is what the brain is solving: If we know the responses of a significant fraction of units from biological networks performing a certain task, we cannot infer the loss function that the brain is optimizing since in principle, numerous different loss functions can have the same/similar minima (Fig. 7 top). In other words, there is typically a many-to-one mapping between loss functions and some point in state space where the functions have a minimum. Some of the different grid models from deep learning and first principles show that this is possible [13, 3, 51, 60, 8]. Conversely, given a reasonable optimization problem that we select based on an organism's ecological niche, we cannot infer a single solution (and thus build truly predictive single-cell tuning models), since there exist several minima to that optimization problem (Fig. 7 bottom). In other words, there is typically a one-to-many mapping from a loss function to its set of solutions. To break this uniqueness problem and arrive at truly predic-

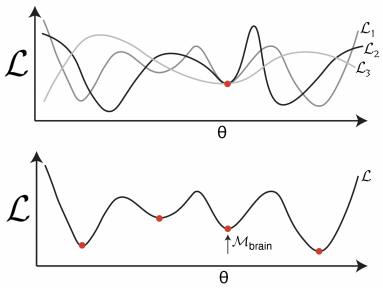

Figure 7: **Challenges in achieving the two central claims of recent DL models of neuroscience**: Top: Building a model that replicates observed neural responses does not guarantee that the loss function used is the brain's objective, as multiple objectives can share a solution. Bottom: Training a network on a plausible loss function or even the correct loss function need not yield the solution the brain has selected because the loss function may have multiple minima, of which the brain selects one based on its constraints, while an ANN selects another, based on the optimization technique used.

tive models or a better understanding of the brain's optimization problems, we must acknowledge, understand, and model the specific inductive biases and constraints present in the biological system we are trying to model. It is untenable to expect success without doing so. This is what we refer to as an informal neural 'No Free Lunch'. These constraints and biases are as informative as other design principles such as the loss function.

What can we learn from DL models about brain circuits without considering and studying biological inductive biases? Population-wide low-dimensional latent representations and dynamics that arise as necessary for solving difficult problems are possibly sufficiently robust and abstracted to be predictive of population dynamics in a neural circuit without addition of detailed biological constraints. Examples include population-level analyses of the visual pathway [31, 4], as well as the population-level low-dimensional dynamics of circuits solving inference tasks [55, 47, 1, 59]. In these cases, the emergent solutions involve fundamental features or latent representations that are necessary for solving the task, and need not be specific to tthe brain; for instance, if the task involves finding pink objects, then there will be representations tuned to pink objects, regardless of whether the system solving the task is neural.

Returning to grid cells, we have shown that they do not generically arise in networks trained to path integrate. Theoretical first-principles work on grid cell representations [18, 53, 39] suggests that the following two factors are important features of the code beyond path integration: a very large coding range and the related property of robustness/intrinsic error correction, both of which have been shown to translate into the problem of packing and maximally separating a large set of coding states within a compact coding space [53]. Adding these properties to the loss is likely to be a more principled and simultaneously more interpretable way to obtain grid-like units rather than by hand-designing a biologically unrealistic DoG or DoS readout. Consistent with this, the very large amount of dropout or other noise required in [3, 13] suggests that the coding-theoretic insight on intrinsic error-correction properties of grid cells and their related large capacity might indeed be central ingredients to produce grid cells.

We hypothesize that key properties of the grid cell code elucidated earlier by theoretical arguments, including but not limited to path integration, may form a sufficient and biologically relevant set for the emergence of grid cells: 1) non-negative activations; 2) equivariant population responses (i.e., the population response always lies on a hypersphere); 3) a path integrating (PI) code or in other words, translation invariant representations [19, 8]; 4) high representational capacity [7, 53, 38]; 5) intrinsic error-correcting capabilities [7, 53]; and finally, 6) uniformly distributed (whitened) and low spatial information per cell, such that the total spatial information of the grid code should be equally distributed across all cells. We suggest that several of these properties (excluding the more grid-cell specific property) are extremely general features or *inductive biases* of neural codes, and could increase the ability of ANN models to make de novo rather than post hoc predictions, thereby removing the need to use synthetic and unrealistic architecture choices (such as DoG and DoS readouts). For recent promising work, see [14].

First-principles continuous attractor network (CAN) models of grid cells made several novel predictions that have been confirmed in experiments [33]: the invariance of cell-cell relationships regardless of environment and behavioral state (constituting, in the terminology of machine learning, "far out of distribution" tests of the models) that define an invariant toroidal attractor manifold [61, 56, 22, 21] and grid-like patterning in the cortical sheet [24]. Deep learning-based models should be held to similar standards.

In sum, the central contribution of scientific interest when building and studying ANN models of the brain that reproduce specific tuning curves is not that the model produced the curves (after all, given the expressive power of deep networks, it is no revelation that training them to generate a given tuning will in fact succeed), but rather to explore and carefully characterize conditions and constraints under which the particular tuning does and does not emerge, to explicitly consider which inductive biases in biology are relevant, and to examine what principles and de novo predictions can be extracted from the models. When not predicting specific tuning curves, but rather population-level response dynamics, the results are expected to reflect the key latent variables required to solve the task. These relatively brain-agnostic features are more robust to specific implementational choices, and thus as predictions more likely to be durable.

9 Acknowledgements

This work was supported by the ONR, the NSF, the Simons Foundation through the Simons Collaboration on the Global Brain, and the Howard Hughes Medical Institute through the Faculty Scholars Program. MK was supported by the MathWorks Science Fellowship.

References

[1] Emergence of dynamically reconfigurable hippocampal responses by learning to perform probabilistic spatial reasoning. *biorxiv*.

[2] High-performing neural network models of visual cortex benefit from high latent dimensionality, July 2022. Pages: 2022.07.13.499969 Section: New Results.

[3] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J. Chadwick, Thomas Degris, Joseph Modayil, Greg Wayne, Hubert Soyer, Fabio Viola, Brian Zhang, Ross Goroshin, Neil Rabinowitz, Razvan Pascanu, Charlie Beattie, Stig Petersen, Amir Sadik, Stephen Gaffney, Helen King, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, and Dharshan Kumaran. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, May 2018.

[4] Pouya Bashivan, Kohitij Kar, and James J. DiCarlo. Neural population control via deep image synthesis. *Science*, 364(6439):eaav9436, May 2019. Publisher: American Association for the Advancement of Science.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020. arXiv: 2005.14165.

[6] Yoram Burak and Ila Fiete. Do We Understand the Emergent Dynamics of Grid Cell Activity? *Journal of Neuroscience*, 26(37):9352–9354, September 2006. Publisher: Society for Neuroscience Section: Journal Club.

[7] Yoram Burak and Ila R Fiete. Unpublished observations. 2008.

[8] Yoram Burak and Ila R. Fiete. Accurate Path Integration in Continuous Attractor Network Models of Grid Cells. *PLOS Computational Biology*, 5(2):e1000291, February 2009. Publisher: Public Library of Science.

[9] Yoram Burak and Ila R Fiete. Accurate path integration in continuous attractor network models of grid cells. *PLoS Comput Biol*, 5(2):e1000291, Feb 2009.

[10] Malcolm G. Campbell, Samuel A. Ocko, Caitlin S. Mallory, Isabel I. C. Low, Surya Ganguli, and Lisa M. Giocomo. Principles governing the integration of landmark and self-motion cues in entorhinal cortical codes for navigation. *Nature Neuroscience*, 21(8):1096–1106, August 2018. Number: 8 Publisher: Nature Publishing Group.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *NIPS Workshop Deep Learning and Representation Learning*, December 2014. Number: arXiv:1412.3555 arXiv:1412.3555 [cs].

[12] Jasmine Collins, Jascha Sohl-Dickstein, and David Sussillo. Capacity and Trainability in Recurrent Neural Networks. *International Conference on Learning Representations*, March 2017. Number: arXiv:1611.09913 arXiv:1611.09913 [cs, stat].

[13] Christopher J Cueva and Xue-Xin Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *International Conference on Learning Representations*, page 19, 2018.

[14] William Dorrell, Peter E Latham, Timothy EJ Behrens, and James CR Whittington. Actionable neural representations: Grid cells from minimal constraints. *arXiv preprint arXiv:2209.15563*, 2022.

[15] Tamir Eliav, Shir R. Maimon, Johnatan Aljadeff, Misha Tsodyks, Gily Ginosar, Liora Las, and Nachum Ulanovsky. Multiscale representation of very large environments in the hippocampus of flying bats. *Science*, 372(6545):eabg4020, May 2021. Publisher: American Association for the Advancement of Science.

[16] Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1402_1.

[17] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO. *arXiv:2005.12729 [cs, stat]*, May 2020. arXiv: 2005.12729.

[18] Ila R Fiete, Yoram Burak, and Ted Brookings. What grid cells convey about rat location. *J Neurosci*, 28(27):6858–71, Jul 2008.

[19] Mark C Fuhs and David S Touretzky. A spin glass model of path integration in rat medial entorhinal cortex. *J Neurosci*, 26(16):4266–4276, 2006.

[20] Marianne Fyhn, Torkel Hafting, Alessandro Treves, May-Britt Moser, and Edvard I Moser. Hippocampal remapping and grid realignment in entorhinal cortex. *Nature*, 446(7132):190–194, 2007.

[21] Richard J. Gardner, Erik Hermansen, Marius Pachitariu, Yoram Burak, Nils A. Baas, Benjamin A. Dunn, May-Britt Moser, and Edvard I. Moser. Toroidal topology of population activity in grid cells. *Nature*, 602(7895):123–128, February 2022. Number: 7895 Publisher: Nature Publishing Group.

[22] Richard J. Gardner, Li Lu, Tanja Wernle, May-Britt Moser, and Edvard I. Moser. Correlation structure of grid cells is preserved during sleep. *Nature Neuroscience*, 22(4):598–608, April 2019. Number: 4 Publisher: Nature Publishing Group.

[23] Joshua I. Glaser, Ari S. Benjamin, Raeed H. Chowdhury, Matthew G. Perich, Lee E. Miller, and Konrad P. Kording. Machine Learning for Neural Decoding. *eNeuro*, 7(4), July 2020. Publisher: Society for Neuroscience Section: Research Article: Methods/New Tools.

[24] Yi Gu, Sam Lewallen, Amina A Kinkhabwala, Cristina Domnisoru, Kijung Yoon, Jeffrey L Gauthier, Ila R Fiete, and David W Tank. A map-like micro-organization of grid cells in the medial entorhinal cortex. *Cell*, 175(3):736–750.e30, 10 2018.

[25] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I. Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, August 2005. Number: 7052 Publisher: Nature Publishing Group.

[26] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6e-RMSProp.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[28] Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A Closer Look at Deep Policy Gradients. *arXiv:1811.02553 [cs, stat]*, May 2020. arXiv: 1811.02553.

[29] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, August 2021. Number: 7873 Publisher: Nature Publishing Group.

[30] I. Kanitscheider and I. R. Fiete. Training recurrent networks to generate hypotheses about how the brain solves hard navigation problems. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[31] Alexander J. E. Kell, Daniel L. K. Yamins, Erica N. Shook, Sam V. Norman-Haignere, and Josh H. McDermott. A Task-Optimized Neural Network Replicates Human Auditory Behavior, Predicts Brain Responses, and Reveals a Cortical Processing Hierarchy. *Neuron*, 98(3):630–644.e16, May 2018.

[32] Mikail Khona, Sarthak Chandra, and Ila R. Fiete. From smooth cortical gradients to discrete modules: spontaneous and topologically robust emergence of modularity in grid cells. *bioRxiv*, page 2021.10.28.466284, January 2022.

[33] Mikail Khona and Ila R Fiete. Attractor and integrator networks in the brain. *preprint at https://arxiv. org/abs/2112.03978*, 2021.

[34] Timothy D. Kim, Thomas Z. Luo, Jonathan W. Pillow, and Carlos D. Brody. Inferring Latent Dynamics Underlying Neural Population Activity via Neural Differential Equations. In *Proceedings of the 38th International Conference on Machine Learning*, pages 5551–5561. PMLR, July 2021. ISSN: 2640-3498.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, January 2017. Number: arXiv:1412.6980 arXiv:1412.6980 [cs].

[36] Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and L. F. Abbott. Optimal Degrees of Synaptic Connectivity. *Neuron*, 93(5):1153–1164.e7, March 2017.

[37] Jesse A. Livezey, Kristofer E. Bouchard, and Edward F. Chang. Deep learning as a tool for neural data analysis: Speech classification and cross-frequency coupling in human sensorimotor cortex. *PLOS Computational Biology*, 15(9):e1007091, September 2019. Publisher: Public Library of Science.

[38] A. Mathis, A. Herz, and M. Stemmler. Optimal population codes for space: grid cells outperform place cells. *Neural Comp.*, 24:2280–2317, 2012.

[39] Alexander Mathis, Andreas V. M. Herz, and Martin Stemmler. Optimal Population Codes for Space: Grid Cells Outperform Place Cells. *Neural Computation*, 24(9):2280–2317, September 2012.

[40] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21(9):1281–1289, September 2018. Number: 9 Publisher: Nature Publishing Group.

[41] Aran Nayebi, Alexander Attinger, Malcolm Campbell, Kiah Hardcastle, Isabel Low, Caitlin S Mallory, Gabriel Mel, Ben Sorscher, Alex H Williams, Surya Ganguli, Lisa Giocomo, and Dan Yamins. Explaining heterogeneity in medial entorhinal cortex with task-driven neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 12167–12179. Curran Associates, Inc., 2021.

[42] J. O'Keefe and J. Dostrovsky. The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34:171–175, 1971. Place: Netherlands Publisher: Elsevier Science.

[43] Talmo D. Pereira, Diego E. Aldarondo, Lindsay Willmore, Mikhail Kislin, Samuel S.-H. Wang, Mala Murthy, and Joshua W. Shaevitz. Fast animal pose estimation using deep neural networks. *Nature Methods*, 16(1):117–125, January 2019. Number: 1 Publisher: Nature Publishing Group.

[44] P. D. Rich, H.-P. Liaw, and A. K. Lee. Large environments reveal the statistical structure governing hippocampal representations. *Science*, 345(6198):814–817, August 2014.

[45] Muhammad Saif-ur Rehman, Robin Lienkämper, Yaroslav Parpaley, Jörg Wellmer, Charles Liu, Brian Lee, Spencer Kellis, Richard Andersen, Ioannis Iossifidis, Tobias Glasmachers, and Christian Klaes. SpikeDeeptector: a deep-learning based method for detection of neural spiking activity. *Journal of Neural Engineering*, 16(5):056003, July 2019. Publisher: IOP Publishing.

[46] Andrew Saxe, Stephanie Nelli, and Christopher Summerfield. If deep learning is the answer, what is the question? *Nature Reviews Neuroscience*, 22(1):55–67, January 2021. Number: 1 Publisher: Nature Publishing Group.

[47] Rylan Schaeffer, Mikail Khona, Leenoy Meshulam, Brain Laboratory International, and Ila Fiete. Reverse-engineering recurrent neural network solutions to a hierarchical inference task for mice. *Advances in Neural Information Processing Systems*, 33:4584–4596, 2020.

[48] Vemund Sigmundson Schøyen, Markus Borud Pettersen, Konstantin Holzhausen, Anders Malthe-Sørensen, and Mikkel Elle Lepperød. Navigating multiple environments with emergent grid cell remapping. *bioRxiv*, 2022.

[49] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. Number: 7587 Publisher: Nature Publishing Group.

[50] Trygve Solstad, Charlotte N. Boccara, Emilio Kropff, May-Britt Moser, and Edvard I. Moser. Representation of Geometric Borders in the Entorhinal Cortex. *Science*, 322(5909):1865–1868, December 2008. Publisher: American Association for the Advancement of Science.

[51] Ben Sorscher, Gabriel C Mel, Surya Ganguli, and Samuel A Ocko. A unified theory for the origin of grid cells through the lens of pattern formation. *Advances in Neural Information Processing Systems*, page 18, 2019.

[52] Ben Sorscher, Gabriel C. Mel, Samuel A. Ocko, Lisa Giocomo, and Surya Ganguli. A unified theory for the computational and mechanistic origins of grid cells. Technical report, bioRxiv, December 2020. Section: New Results Type: article.

[53] Sameet Sreenivasan and Ila Fiete. Grid cells generate an analog error-correcting code for singularly precise neural computation. *Nat Neurosci*, 14(10):1330–7, Sep 2011.

[54] Hanne Stensola, Tor Stensola, Trygve Solstad, Kristian Frøland, May-Britt Moser, and Edvard I. Moser. The entorhinal grid map is discretized. *Nature*, 492(7427):72–78, December 2012. Number: 7427 Publisher: Nature Publishing Group.

[55] David Sussillo and Omri Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 25(3):626–649, March 2013.

[56] S.G. Trettel, J.B. Trimper, E. Hwaun, I.R. Fiete, and L.L. Colgin. Grid cell co-activity patterns during sleep reflect spatial overlap of grid fields during active behaviors. *Nat Neurosci*, 22(4):609–617, 04 2019.

[57] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E. Turner, Zoubin Ghahramani, and Sergey Levine. The Mirage of Action-Dependent Baselines in Reinforcement Learning. *arXiv:1802.10031 [cs, stat]*, November 2018. arXiv: 1802.10031.

[58] Greta Tuckute, Jenelle Feather, Dana Boebinger, and Josh H McDermott. Many but not all deep neural network audio models capture brain responses and exhibit hierarchical region correspondence. *bioRxiv*, 2022.

[59] Jakob Voigts, Ingmar Kanitscheider, Nicholas J. Miller, Enrique H. S. Toloza, Jonathan P. Newman, Ila R. Fiete, and Mark T. Harnett. Spatial reasoning via recurrent neural dynamics in mouse retrosplenial cortex. *biorxiv*, April 2022.

[60] James C. R. Whittington, Timothy H. Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy E. J. Behrens. The Tolman-Eichenbaum Machine: Unifying Space and Relational Memory through Generalization in the Hippocampal Formation. *Cell*, 183(5):1249–1263.e23, November 2020.

[61] K.J. Yoon, M.A. Buice, R. Barry, C.and Hayman, N. Burgess, and I.R. Fiete. Specific evidence of low-dimensional continuous attractor dynamics in grid cells. *Nat Neurosci*, 16(8):1077–84, Aug 2013.

10 Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not feel our paper has potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We haven't yet had time to collect this information, but we will add this information to the final version if accepted.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Position Encodings

Suppose we sample a sequence of positions $x_0, ..., x_T \in \mathbb{R}^2$ and a sequence of velocities $v_1, ..., v_T \in \mathbb{R}^2$, where $x_t = x_{t-1} + v_t$. We want to train the networks in a supervised manner to predict (a possible encoding of) their position. We used the below encodings as different regression targets. For some encodings that required place cell populations, we denote the number of place cells N_p and denote their locations $\{p_i\}_{i=1}^{N_p}$, sampled uniformly at random within the 2.2 m $\times$ 2.2 m arena.

- **"Low-dimensional" Cartesian:** x_t
- **"High-dimensional" Cartesian:** Let $A \in \mathbb{R}^{N_p \times 2}, b \in \mathbb{R}^{N_p}$ have entries sampled i.i.d. from $Uniform(-1, 1)$. The target is a vector in $\mathbb{R}^{N_p}$ given by:
$$Ax_t + b$$
- **Polar:** Let (r_t, θ_t) denote the polar form of the agent's position x_t. The target is a vector in $\mathbb{R}^{N_p}$, half comprised of "distance" units and half comprised of "direction" units. Let $A \in \mathbb{R}^{0.5N_p \times 1}, b \in \mathbb{R}^{0.5N_p}$ have entries sampled i.i.d. from $Uniform(-1, 1)$; the distance cells have activites:
$$Ar_t + b$$
Let $\mu \in [-\pi, \pi]^{N_p/2}$ have entries sampled i.i.d. uniformly at random. The direction cells have entries given by von-Mises-like bumps:
$$\exp\Big(\cos(\theta_t - \mu_i)\Big)$$
- **Gaussian:** A vector in $\mathbb{R}^{N_p}$ whose entries are given by:
$$\exp\Big(-\frac{1}{2\sigma_E^2}||x_t - p_i||^2\Big)$$
- **Difference of Gaussians:** A vector in $\mathbb{R}^{N_p}$ whose entries are given by:
$$\exp\Big(-\frac{1}{2\sigma_E^2}||x_t - p_i||^2\Big) - \exp\Big(-\frac{1}{2\sigma_I^2}||x_t - p_i||^2\Big)$$
- **Difference of Softmaxes:** A vector in $\mathbb{R}^{N_p}$ whose entries are given by:
$$Softmax\Big(-\frac{1}{2\sigma_E^2}||x_t - p_i||^2\Big) - Softmax\Big(-\frac{1}{2\sigma_I^2}||x_t - p_i||^2\Big)$$
- **Softmax of Differences:** A vector in $\mathbb{R}^{N_p}$ whose entries are given by:
$$Softmax\Big(-\frac{1}{2\sigma_E^2}||x_t - p_i||^2 + \frac{1}{2\sigma_I^2}||x_t - p_i||^2\Big)$$

B Grid Scores and Grid Cell Thresholds

What qualifies as a grid cell? The most commonly used method of quantifying grid cells is via the "grid score", which functions by binning neural activity into rate maps using spatial position, applying an adaptive smoother, then taking a circular sample of the autocorrelation centered on the central peak and comparing it to rotated versions of the same circular sample. The 60° grid score is specifically given by:
$$(corr[60] + corr[120])/2 - (corr[30] + corr[90] + corr[150])/3$$

We used the grid scorer implementation used by [3] (https://github.com/deepmind/grid-cells/blob/master/scores.py), [51] (https://github.com/ganguli-lab/grid-pattern-formation/blob/master/scores.py) and [41].

What score is sufficient to qualify as a grid score? Experimentalists have used thresholds of 0.3 [50] and 0.349 [10] on biological neurons, whereas computationalists have used 0.3 [41] and 0.37 [3, 51] on artificial neurons. We found that for artificial neurons, these thresholds are far too low (Fig. 8); ANN units with grid scores > 0.4, and even as high as 1.3, often look nothing like grid cells. This is because the grid score looks for $60°$ rotational symmetry, and while grid cells are indeed symmetric, so are many other rate maps. Private conversations with the authors of [3, 51, 41] showed everyone was in agreement that grid scores are a metric in need of improvement. After internal disagreement, we compromised at a grid score threshold of 0.8.

C Number of Bins for Computing Rate Maps

The first step in computing grid scores is determining the number of bins to use to compute rate maps. However, establishing the number of bins used by previous approaches proved baffling difficult. The original experimental work used 5 cm by 5 cm bins [25]. Since the square arena used in these experiments is 2.2 m by 2.2 m (same as [3, 51, 41], the number of bins should be 44 x 44. However, we found discrepancies in previous approaches' text and code.

[3]'s text claims to use 32 x 32 bins of 6.875 cm x 6.875 cm ("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."), but their code [2] used 20 x 20 bins of 11 cm x 11 cm. [51] claimed to use 2 cm x 2 cm bins (""Grid score was evaluated as in [3]]. A spatial ratemap was computed for each neuron by binning the agent's position into 2cm x 2cm bins, and computing the average firing rate within each bin.") but their code [3] similarly used 20 x 20 bins. [41] claimed to use 5 cm x 5 cm bins ("Nayebi: "We bin the positions in each environment using 5 cm bins, following prior work [Hardcastle et al., 2017, Butler et al., 2019, Low et al., 2020]. Thus, the 100cm^2 environment used 400 (20×20) bins, the 150cm^2 environment used 900 (30×30) bins, and the 400cm 1D track used

[2] https://github.com/deepmind/grid-cells/blob/master/train.py#L201

[3] https://github.com/ganguli-lab/grid-pattern-formation/blob/master/visualize.py#L136

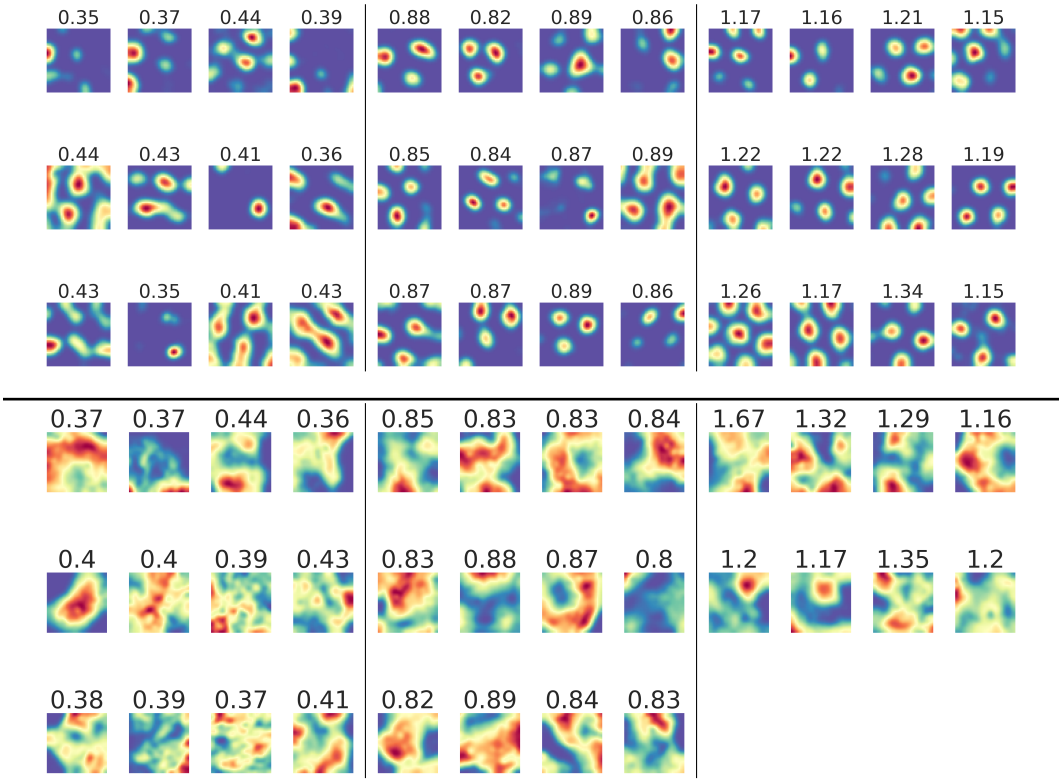

Figure 8: **Grid scoring is the dominant method to identify grid cells, but we found it performs extremely poorly at identifying grid cells.** Top: example rate maps from low position decoding error Difference-of-Softmaxes (DoS) networks three grid score ranges: $[0.35, 0.45)$, $[0.8, 0.9)$, $[1.15, \infty)$. Bottom: example rate maps from low position decoding error Difference-of-Gaussians (DoG) networks. We considered three grid score thresholds: 0.3 (used by some experimentalists), 0.8 (low probability of finding grid cells), 1.15 (decent probability of finding grid cells).

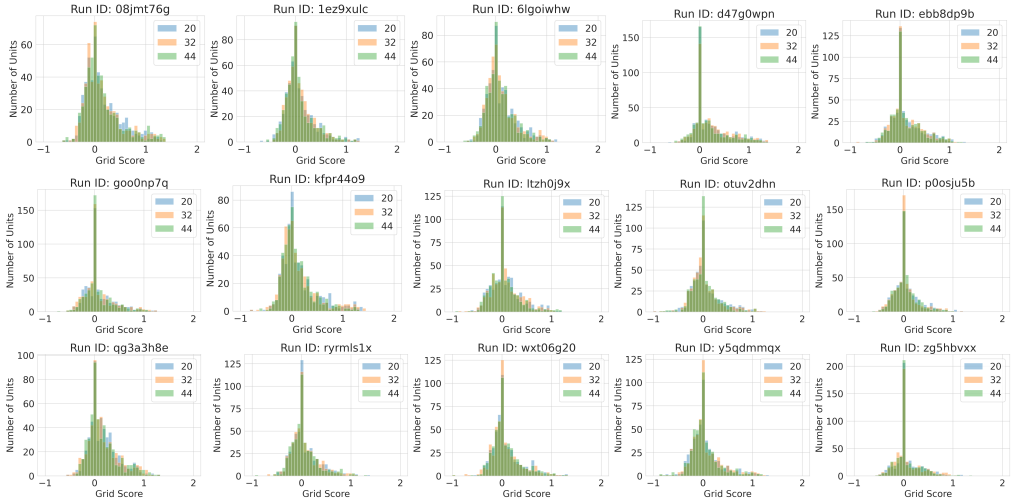

Figure 9: Grid score distributions do not differ as a function of number of bins: 400 (20 x 20; blue), 1024 (32 x 32; orange), 1936 (44 x 44; green).

80 bins.") but their code similarly used 20 x 20 bins. The code suggests that [41] used the grid scorer of [51], who in turn used the grid scorer of [3].

Consequently, we tested what effect the number of bins has on the distribution of grid scores. We found that the number of bins appears to have little to no effect (Fig. 9), so we chose to use 44 x 44 bins since this yields bins of size 5 cm x 5 cm.

D Place cell autocorrelation

In this section, we will derive the form of the place cell correlation function, centered at $\vec{\mu}$. When the spatial tuning curve is a difference-of-Gaussians, the correlation function is *also* a Difference-of-Gaussians, albeit with different parameters. Consider the spatial tuning curve:

$$P(\mathbf{x};\vec{\mu}) = \alpha_E \exp\Big(-\frac{(\mathbf{x}-\vec{\mu})^2}{2\sigma_E^2}\Big) - \alpha_I \exp\Big(-\frac{(\mathbf{x}-\vec{\mu})^2}{2\sigma_I^2}\Big)$$

So the correlation between place cells centered at $\vec{\mu_1}$ and $\vec{\mu_2}$ with $\mathbf{r} = \vec{\mu_1} - \vec{\mu_2}$ is given by

$$C_{\mu_1,\mu_2} = \int P(\mathbf{x};\mu_1)P(\mathbf{x};\mu_2)d\mathbf{x} = \int P(\mathbf{x};0)P(\mathbf{x};\mathbf{r})d\mathbf{x}$$

Simplifying the above expression, using the identity: $\int dx\mathcal{N}(x;\mu_f,\sigma_f)\mathcal{N}(r-x;\mu_f,\sigma_f) = \mathcal{N}(r;\mu_f+\mu_g,\sqrt{\sigma_f^2+\sigma_g^2})$, we get:

$$C_{\mu_1,\mu_2} = \alpha_E^2 \exp\Big(\frac{-\mathbf{r}^2}{4\sigma_E^2}\Big) + \alpha_I^2 \exp\Big(\frac{-\mathbf{r}^2}{4\sigma_I^2}\Big) - 2\alpha_E\alpha_I \exp\Big(\frac{-\mathbf{r}^2}{2(\sigma_E^2+\sigma_I^2)}\Big)$$

E Sweeps

E.1 Cartesian (Low Dimensional)


```yaml
method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
    values:
      - 1024
  Np:
    values:
      - 2
  activation:
    values:
      - relu
      - tanh
      - sigmoid
      - linear
  batch_size:
    values:
      - 200
  bin_side_in_m:
    values:
      - 0.05
  box_height_in_m:
    values:
      - 2.2
  box_width_in_m:
    values:
      - 2.2
  initializer:
    values:
      - glorot_uniform
      - glorot_normal
      - orthogonal
  is_periodic:
    values:
      - false
  learning_rate:
    values:
      - 0.0001
  n_epochs:
    values:
      - 20
  n_grad_steps_per_epoch:
    values:
      - 10000
  n_place_fields_per_cell:
    values:
      - 1
  optimizer:
    values:
      - sgd
      - adam
      - rmsprop
  place_cell_rf:
```

```
    values:
      - 0
  place_field_loss:
    values:
      - mse
  place_field_normalization:
    values:
      - none
  place_field_values:
    values:
      - cartesian
  readout_dropout:
    values:
      - 0
      - 0.5
  recurrent_dropout:
    values:
      - 0
      - 0.5
  rnn_type:
    values:
      - RNN
      - LSTM
      - UGRNN
      - GRU
  seed:
    values:
      - 0
      - 1
      - 2
  sequence_length:
    values:
      - 20
  surround_scale:
    values:
      - 1
  weight_decay:
    values:
      - 0
      - 0.0001
```


E.2 Cartesian (Low Dimensional, Random)


```
method: random
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  activation:
    values: [
      'relu',
      'tanh',
      'sigmoid',
      'linear',
    ]
  batch_size:
    values: [30, 60, 90, 120, 150, 180]
  bin_side_in_m:
    values: [0.05]
```

box_height_in_m:
  values: [2.2]
box_width_in_m:
  values: [2.2]
initializer:
  values: [
    'glorot_uniform',
    'glorot_normal',
    'orthogonal',
  ]
is_periodic:
  values: [False]
learning_rate:
  values: [0.005, 0.001, 0.0005, 0.0001]
n_epochs:
  values: [1, 4, 8, 12, 16]
n_grad_steps_per_epoch:
  values: [10000]
n_place_fields_per_cell:
  values: [1]
Ng:
  values: [1024]
Np:
  values: [2]
optimizer:
  values: [
      'adam',
      'rmsprop',
  ]
place_field_loss:
  values: [
    'mse',
  ]
place_field_values:
  values: [
    'cartesian',
  ]
place_field_normalization:
  values: [
    'none',
  ]
place_cell_rf:
  values: [
    0.
  ]
readout_dropout:
  values: [0., 0.05, 0.1, 0.2, 0.5]
recurrent_dropout:
  values: [0., 0.05, 0.1, 0.2, 0.5]
rnn_type:
  values: [
    'RNN',
    'LSTM',
    'UGRNN',
    'GRU',
  ]
seed:
  values: [0, 1, 2]
sequence_length:

```
    values: [20, 25, 30, 35, 40]
  surround_scale:
    values: [1.]
  weight_decay:
      values: [0., 0.0001, 0.0005, 0.001, 0.005, 0.01 ]
```


E.3 Cartesian (High Dimensional)


```
method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
    values:
      - 1024
  Np:
    values:
      - 512
  activation:
    values:
      - relu
      - tanh
      - sigmoid
      - linear
  batch_size:
    values:
      - 200
  bin_side_in_m:
    values:
      - 0.05
  box_height_in_m:
    values:
      - 2.2
  box_width_in_m:
    values:
      - 2.2
  initializer:
    values:
      - glorot_uniform
      - glorot_normal
      - orthogonal
  is_periodic:
    values:
      - false
  learning_rate:
    values:
      - 0.0001
  n_epochs:
    values:
      - 20
  n_grad_steps_per_epoch:
    values:
      - 1000
  n_place_fields_per_cell:
    values:
      - 1
  optimizer:
    values:
```

```
      - adam
  place_cell_rf:
    values:
      - 0
  place_field_loss:
    values:
      - mse
  place_field_normalization:
    values:
      - none
  place_field_values:
    values:
      - high_dim_cartesian
  readout_dropout:
    values:
      - 0
      - 0.5
  recurrent_dropout:
    values:
      - 0
      - 0.5
  rnn_type:
    values:
      - RNN
      - LSTM
      - UGRNN
      - GRU
  seed:
    values:
      - 0
      - 1
      - 2
  sequence_length:
    values:
      - 20
  surround_scale:
    values:
      - 0
  weight_decay:
    values:
      - 0
      - 0.0001
```


E.4 Cartesian (High Dimensional, Random)


```
method: random
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  activation:
    values: [
      'relu',
      'tanh',
      'sigmoid',
      'linear',
    ]
  batch_size:
    values: [30, 60, 90, 120, 150, 180]
```

```
bin_side_in_m:
  values: [0.05]
box_height_in_m:
  values: [2.2]
box_width_in_m:
  values: [2.2]
initializer:
  values: [
    'glorot_uniform',
    'glorot_normal',
    'orthogonal',
  ]
is_periodic:
  values: [False]
learning_rate:
  values: [0.005, 0.001, 0.0005, 0.0001]
n_epochs:
  values: [1, 4, 8, 12, 16]
n_grad_steps_per_epoch:
  values: [1000]
n_place_fields_per_cell:
  values: [ 1 ]
Ng:
  values: [1024]
Np:
  values: [128, 256, 512]
optimizer:
  values: [
    'adam',
    'rmsprop',
  ]
place_field_loss:
  values: [
    'mse',
  ]
place_field_values:
  values: [
    'high_dim_cartesian'
  ]
place_field_normalization:
  values: [
    'none',
  ]
place_cell_rf:
  values: [
    0.0,
  ]
recurrent_dropout:
  values: [ 0., 0.05, 0.1, 0.2, 0.5  ]
readout_dropout:
  values: [ 0., 0.05, 0.1, 0.2, 0.5 ]
rnn_type:
  values: [
    'RNN',
    'LSTM',
    'UGRNN',
    'GRU',
  ]
seed:
```

```
    values: [ 0, 1, 2 ]
  sequence_length:
    values: [20, 25, 30, 35, 40]
  surround_scale:
    values: [
      0.
    ]
  weight_decay:
      values: [0., 0.0001, 0.0005, 0.001, 0.005, 0.01   ]
```


E.5 Polar (High Dimensional)


```
method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
    values:
      - 1024
  Np:
    values:
      - 512
  activation:
    values:
      - relu
      - tanh
      - linear
      - sigmoid
  batch_size:
    values:
      - 200
  bin_side_in_m:
    values:
      - 0.05
  box_height_in_m:
    values:
      - 2.2
  box_width_in_m:
    values:
      - 2.2
  initializer:
    values:
      - glorot_uniform
      - glorot_normal
      - orthogonal
  is_periodic:
    values:
      - false
  learning_rate:
    values:
      - 0.0001
  n_epochs:
    values:
      - 20
  n_grad_steps_per_epoch:
    values:
      - 1000
  n_place_fields_per_cell:
```

```
    values:
      - 1
  optimizer:
    values:
      - adam
  place_cell_rf:
    values:
      - 0
  place_field_loss:
    values:
      - mse
  place_field_normalization:
    values:
      - none
  place_field_values:
    values:
      - high_dim_polar
  readout_dropout:
    values:
      - 0
      - 0.5
  recurrent_dropout:
    values:
      - 0
      - 0.5
  rnn_type:
    values:
      - RNN
      - LSTM
      - UGRNN
      - GRU
  seed:
    values:
      - 0
      - 1
      - 2
  sequence_length:
    values:
      - 20
  surround_scale:
    values:
      - 0
  weight_decay:
    values:
      - 0
      - 0.0001
```


E.6 Polar (High Dimensional, Random)


```
method: random
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  activation:
    values: [
        'relu',
        'tanh',
        'linear',
```

```
        'sigmoid',
    ]
  batch_size:
    values: [30, 60, 90, 120, 150, 180]
  bin_side_in_m:
    values: [0.05]
  box_height_in_m:
    values: [2.2]
  box_width_in_m:
    values: [2.2]
  initializer:
    values: [
      'glorot_uniform',
      'glorot_normal',
      'orthogonal',
    ]
  is_periodic:
    values: [False]
  learning_rate:
    values: [0.005, 0.001, 0.0005, 0.0001]
  n_epochs:
    values: [1, 4, 8, 12, 16]
  n_grad_steps_per_epoch:
    values: [5000]
  n_place_fields_per_cell:
    values: [ 1 ]
  Ng:
    values: [1024]
  Np:
    values: [128, 256, 512]
  optimizer:
    values: [
      'adam',
      'rmsprop',
    ]
  place_field_loss:
    values: [
      'mse',
    ]
  place_field_values:
    values: [
      'high_dim_polar',
    ]
  place_field_normalization:
    values: [
      'none',
    ]
  place_cell_rf:
    values: [
      0.
    ]
  readout_dropout:
    values: [0., 0.05, 0.1, 0.2, 0.5 ]
  recurrent_dropout:
    values: [0., 0.05, 0.1, 0.2, 0.5 ]
  rnn_type:
    values: [
        'RNN',
        'LSTM',
```

```
        'UGRNN',
        'GRU',
    ]
  seed:
    values: [ 0, 1, 2 ]
  sequence_length:
    values: [20, 25, 30, 35, 40]
  surround_scale:
    values: [0.]
  weight_decay:
      values: [0., 0.0001, 0.0005, 0.001, 0.005, 0.01 ]
```


E.7 Gaussian Place Cells


```
method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
    values:
      - 1024
  Np:
    values:
      - 512
  activation:
    values:
      - linear
      - relu
      - tanh
      - sigmoid
  batch_size:
    values:
      - 200
  bin_side_in_m:
    values:
      - 0.05
  box_height_in_m:
    values:
      - 2.2
  box_width_in_m:
    values:
      - 2.2
  initializer:
    values:
      - glorot_uniform
  is_periodic:
    values:
      - false
  learning_rate:
    values:
      - 0.0001
  n_epochs:
    values:
      - 20
  n_grad_steps_per_epoch:
    values:
      - 10000
  n_place_fields_per_cell:
```

```
    values:
      - 1
  optimizer:
    values:
      - adam
  place_cell_rf:
    values:
      - 0.08
      - 0.1
      - 0.12
      - 0.14
      - 0.16
      - 0.2
      - 0.24
      - 0.28
  place_field_loss:
    values:
      - crossentropy
  place_field_normalization:
    values:
      - global
  place_field_values:
    values:
      - gaussian
  readout_dropout:
    values:
      - 0
      - 0.5
  recurrent_dropout:
    values:
      - 0
      - 0.5
  rnn_type:
    values:
      - RNN
      - LSTM
      - UGRNN
      - GRU
  seed:
    values:
      - 0
      - 1
  sequence_length:
    values:
      - 20
  surround_scale:
    values:
      - 1
  weight_decay:
    values:
      - 0.0001
```


E.8 Gaussian Place Cells (Random)


```
method: random
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
```

```
activation:
  values: [
      'linear',
      'relu',
      'tanh',
      'sigmoid',
  ]
batch_size:
  values: [30, 60, 90, 120, 150, 180]
bin_side_in_m:
  values: [0.05]
box_height_in_m:
  values: [2.2]
box_width_in_m:
  values: [2.2]
initializer:
  values: [
    'glorot_uniform',
    'glorot_normal',
    'orthogonal',
  ]
is_periodic:
  values: [False]
learning_rate:
  values: [0.005, 0.001, 0.0005, 0.0001]
n_epochs:
  values: [1, 4, 8, 12, 16]
n_grad_steps_per_epoch:
  values: [10000]
n_place_fields_per_cell:
  values: [
    1.0,
  ]
Ng:
  values: [1024]
Np:
  values: [128, 256, 512]
optimizer:
  values: [
    'adam',
    'rmsprop'
  ]
place_field_loss:
  values: [
    'crossentropy',
  ]
place_field_values:
  values: [
    'gaussian',
  ]
place_field_normalization:
  values: [
    'global',
  ]
place_cell_rf:
  values: [
    0.08,
    0.10,
    0.12,
```

```
      0.14,
      0.16,
      0.20,
      0.24,
      0.28,
      0.32,
      0.36,
      0.40,
    ]
  readout_dropout:
    values: [ 0., 0.05, 0.1, 0.2, 0.5  ]
  recurrent_dropout:
    values: [ 0., 0.05, 0.1, 0.2, 0.5  ]
  rnn_type:
    values: [
      'RNN',
      'LSTM',
      'UGRNN',
      'GRU',
    ]
  seed:
    values: [ 0, 1, 2]
  sequence_length:
    values: [20, 25, 30, 35, 40]
  surround_scale:
    values: [1.]
  weight_decay:
      values: [0., 0.0001, 0.0005, 0.001, 0.005, 0.01  ]
```


E.9 Difference-of-Gaussians Place Cells


```
method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
    values:
      - 1024
  Np:
    values:
      - 512
  activation:
    values:
      - relu
  batch_size:
    values:
      - 200
  bin_side_in_m:
    values:
      - 0.05
  box_height_in_m:
    values:
      - 2.2
  box_width_in_m:
    values:
      - 2.2
  initializer:
    values:
```

```
    - glorot_uniform
is_periodic:
  values:
    - false
learning_rate:
  values:
    - 0.0001
n_epochs:
  values:
    - 20
n_grad_steps_per_epoch:
  values:
    - 10000
n_place_fields_per_cell:
  values:
    - 1
optimizer:
  values:
    - adam
place_cell_rf:
  values:
    - 0.05
    - 0.1
    - 0.15
    - 0.2
    - 0.3
    - 0.4
    - 0.5
place_field_loss:
  values:
    - crossentropy
place_field_normalization:
  values:
    - global
place_field_values:
  values:
    - true_difference_of_gaussians
readout_dropout:
  values:
    - 0
    - 0.5
recurrent_dropout:
  values:
    - 0
    - 0.5
rnn_type:
  values:
    - RNN
    - LSTM
    - UGRNN
    - GRU
seed:
  values:
    - 0
    - 1
    - 2
sequence_length:
  values:
    - 20
```

```
  surround_scale:
    values:
      - 1.5
      - 2
      - 2.5
      - 3
      - 4
      - 5
      - 6
  weight_decay:
    values:
      - 0.0001
```


E.10 Difference-of-Gaussians Place Cells (Random)


```
method: random
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  activation:
    values: [
      'linear',
      'relu',
      'tanh',
      'sigmoid',
    ]
  batch_size:
    values: [30, 60, 90, 120, 150, 180]
  bin_side_in_m:
    values: [0.05]
  box_height_in_m:
    values: [2.2]
  box_width_in_m:
    values: [2.2]
  initializer:
    values: [
      'glorot_uniform',
      'glorot_normal',
      'orthogonal',
    ]
  is_periodic:
    values: [False]
  learning_rate:
    values: [0.005, 0.001, 0.0005, 0.0001]
  n_epochs:
    values: [1, 4, 8, 12, 16]
  n_grad_steps_per_epoch:
    values: [10000]
  n_place_fields_per_cell:
    values: [ 1 ]
  Ng:
    values: [1024]
  Np:
    values: [128, 256, 512]
  optimizer:
    values: [
      'adam',
      'rmsprop'
```

```
    ]
  place_field_loss:
    values: [
      'crossentropy',
    ]
  place_field_values:
    values: [
      'true_difference_of_gaussians'
    ]
  place_field_normalization:
    values: [
      'global',
    ]
  place_cell_rf:
    values: [
      0.08,
      0.10,
      0.12,
      0.14,
      0.16,
      0.20,
      0.24,
      0.28,
      0.32,
      0.36,
      0.40,
    ]
  recurrent_dropout:
    values: [ 0., 0.05, 0.1, 0.2, 0.5 ]
  readout_dropout:
    values: [ 0., 0.05, 0.1, 0.2, 0.5 ]
  rnn_type:
    values: [
      'RNN',
      'LSTM',
      'UGRNN',
      'GRU',
    ]
  seed:
    values: [ 0, 1, 2 ]
  sequence_length:
    values: [20, 25, 30, 35, 40]
  surround_scale:
    values: [
      1.5,
      2.,
      2.5,
      3.,
      3.5,
      4.,
      4.5,
      5.,
      5.5,
      6.,
    ]
  weight_decay:
      values: [ 0., 0.0001, 0.0005, 0.001, 0.005, 0.01 ]
```

E.11 Difference-of-Softmaxes Place Cells


```yaml
method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
    values:
      - 1024
  Np:
    values:
      - 512
  activation:
    values:
      - relu
      - tanh
  batch_size:
    values:
      - 200
  bin_side_in_m:
    values:
      - 0.05
  box_height_in_m:
    values:
      - 2.2
  box_width_in_m:
    values:
      - 2.2
  initializer:
    values:
      - glorot_uniform
  is_periodic:
    values:
      - false
  learning_rate:
    values:
      - 0.0001
  n_epochs:
    values:
      - 20
  n_grad_steps_per_epoch:
    values:
      - 10000
  n_place_fields_per_cell:
    values:
      - 1
  optimizer:
    values:
      - adam
  place_cell_rf:
    values:
      - 0.08
      - 0.09
      - 0.1
      - 0.11
      - 0.12
      - 0.13
      - 0.14
```

```
      - 0.15
      - 0.16
      - 0.17
      - 0.18
      - 0.19
      - 0.2
      - 0.24
      - 0.28
      - 0.32
  place_field_loss:
    values:
      - crossentropy
  place_field_normalization:
    values:
      - global
  place_field_values:
    values:
      - difference_of_gaussians
  readout_dropout:
    values:
      - 0
      - 0.5
  recurrent_dropout:
    values:
      - 0
      - 0.5
  rnn_type:
    values:
      - RNN
      - LSTM
      - UGRNN
      - GRU
  seed:
    values:
      - 0
      - 1
      - 2
  sequence_length:
    values:
      - 20
  surround_scale:
    values:
      - 1.5
      - 2
      - 2.5
      - 3
      - 4
  weight_decay:
    values:
      - 0.0001
```


E.12 Difference-of-Softmaxes Place Cells (Random)


```
method: random
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  activation:
```

```
    values: [
      'linear',
      'relu',
      'tanh',
      'sigmoid',
    ]
  batch_size:
    values: [30, 60, 90, 120, 150, 180]
  bin_side_in_m:
    values: [0.05]
  box_height_in_m:
    values: [2.2]
  box_width_in_m:
    values: [2.2]
  initializer:
    values: [
      'glorot_uniform',
      'glorot_normal',
      'orthogonal',
    ]
  is_periodic:
    values: [False]
  learning_rate:
    values: [0.005, 0.001, 0.0005, 0.0001]
  n_epochs:
    values: [20]
  n_grad_steps_per_epoch:
    values: [10000]
  n_place_fields_per_cell:
    values: [
      1.0,
    ]
  Ng:
    values: [1024]
  Np:
    values: [128, 256, 512]
  optimizer:
    values: [
      'adam',
      'rmsprop',
    ]
  place_field_loss:
    values: [
      'crossentropy',
    ]
  place_field_values:
    values: [
      'difference_of_gaussians',
    ]
  place_field_normalization:
    values: [
      'global',
    ]
  place_cell_rf:
    values: [
      0.08,
      0.10,
      0.12,
      0.14,
```

```
      0.16,
      0.20,
      0.24,
      0.28,
      0.32,
      0.36,
      0.40,
      0.44,
      0.48,
    ]
  readout_dropout:
    values: [0., 0.05, 0.1, 0.2, 0.5]
  recurrent_dropout:
    values: [0., 0.05, 0.1, 0.2, 0.5]
  rnn_type:
    values: [
      'RNN',
      'LSTM',
      'UGRNN',
      'GRU',
    ]
  seed:
    values: [ 0, 1, 2 ]
  sequence_length:
    values: [20, 25, 30, 35, 40]
  surround_scale:
    values: [
      1.5,
      2.,
      2.5,
      3.0,
      3.5,
      4.0,
      4.5,
      5.0,
      5.5,
      6.0,
    ]
  weight_decay:
      values: [0., 0.0001, 0.0005, 0.001, 0.005, 0.01 ]
```


E.13 Softmax-of-Differences Place Cells


```
method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
    values:
      - 1024
  Np:
    values:
      - 512
  activation:
    values:
      - relu
  batch_size:
    values:
```

```
    - 200
bin_side_in_m:
  values:
    - 0.05
box_height_in_m:
  values:
    - 2.2
box_width_in_m:
  values:
    - 2.2
initializer:
  values:
    - glorot_uniform
is_periodic:
  values:
    - false
learning_rate:
  values:
    - 0.0001
n_epochs:
  values:
    - 20
n_grad_steps_per_epoch:
  values:
    - 10000
n_place_fields_per_cell:
  values:
    - 1
optimizer:
  values:
    - adam
place_cell_rf:
  values:
    - 0.09
    - 0.12
    - 0.15
    - 0.18
    - 0.21
    - 0.24
place_field_loss:
  values:
    - crossentropy
place_field_normalization:
  values:
    - global
place_field_values:
  values:
    - softmax_of_differences
readout_dropout:
  values:
    - 0
recurrent_dropout:
  values:
    - 0
rnn_type:
  values:
    - RNN
    - LSTM
    - UGRNN
```

```
      - GRU
  seed:
    values:
      - 0
      - 1
      - 2
  sequence_length:
    values:
      - 20
  surround_scale:
    values:
      - 1.5
      - 2
      - 2.5
      - 3
  weight_decay:
    values:
      - 0.0001
```


E.14 Multiple Scales and Multiple Fields Difference-of-Softmaxes Place Cells


```
method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
    values:
      - 1024
  Np:
    values:
      - 512
  activation:
    values:
      - relu
  batch_size:
    values:
      - 200
  bin_side_in_m:
    values:
      - 0.05
  box_height_in_m:
    values:
      - 2.2
  box_width_in_m:
    values:
      - 2.2
  initializer:
    values:
      - glorot_uniform
  is_periodic:
    values:
      - false
  learning_rate:
    values:
      - 0.0001
  n_epochs:
    values:
      - 20
```

```
  n_grad_steps_per_epoch:
    values:
      - 10000
  n_place_fields_per_cell:
    values:
      - Poisson( 2.0 )
      - Poisson( 3.0 )
  optimizer:
    values:
      - adam
  place_cell_rf:
    values:
      - 0.12
      - Uniform( 0.06 , 0.24 )
      - Uniform( 0.06 , 1.0 )
  place_field_loss:
    values:
      - crossentropy
  place_field_normalization:
    values:
      - global
  place_field_values:
    values:
      - difference_of_gaussians
  readout_dropout:
    values:
      - 0
  recurrent_dropout:
    values:
      - 0
  rnn_type:
    values:
      - RNN
      - UGRNN
  seed:
    values:
      - 0
      - 1
      - 2
  sequence_length:
    values:
      - 20
  surround_scale:
    values:
      - 2
      - Uniform( 1.50 , 2.50 )
      - Uniform( 1.25 , 4.50 )
  weight_decay:
    values:
      - 0.0001
```


E.15 Nayebi et al. 2021 [41] Replication


```
method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
```

```
  values:
    - 4096
Np:
  values:
    - 512
activation:
  values:
    - relu
batch_size:
  values:
    - 200
bin_side_in_m:
  values:
    - 0.05
box_height_in_m:
  values:
    - 2.2
box_width_in_m:
  values:
    - 2.2
initializer:
  values:
    - glorot_uniform
is_periodic:
  values:
    - false
learning_rate:
  values:
    - 0.0001
n_epochs:
  values:
    - 20
n_grad_steps_per_epoch:
  values:
    - 10000
n_place_fields_per_cell:
  values:
    - 1
optimizer:
  values:
    - adam
place_cell_rf:
  values:
    - 0.12
place_field_loss:
  values:
    - crossentropy
place_field_normalization:
  values:
    - global
place_field_values:
  values:
    - difference_of_gaussians
readout_dropout:
  values:
    - 0
recurrent_dropout:
  values:
    - 0
```

```yaml
rnn_type:
  values:
    - RNN
    - LSTM
    - UGRNN
    - GRU
    - SRNN
seed:
  values:
    - 0
    - 1
    - 2
    - 3
    - 4
sequence_length:
  values:
    - 20
surround_scale:
  values:
    - 2
weight_decay:
  values:
    - 0.0001
```