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

Rylan Schaeffer 1 2  Mikail Khona 3 4  Ila Rani Fiete 2 4

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 often gets neither. We begin by reviewing the principles of grid cell mechanism and function obtained from analytical and 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. Finally, 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. In conclusion, caution and consideration, together with biological knowledge, are warranted in building and interpreting deep learning models in Neuroscience.

1. Introduction

Within the past decade, deep learning (DL) has leapt from obscurity to underpinning nearly every success story in machine learning, e.g., Mnih et al. (2015); Silver et al. (2016); Brown et al. (2020) and increasingly many advances in fundamental science research, e.g., Jumper et al. (2021); Davies et al. (2021); Bellemare et al. (2020). In neuroscience, deep learning is similarly gaining widespread adoption as a useful method for behavioral and neural data analysis (Sussillo et al., 2021; Davies et al., 2016; Saif-ur Rehman et al., 2019; Luxem et al., 2020; Pereira et al., 2019; Glaser et al., 2020; Livezey et al., 2019; Kim et al., 2021; Mathis et al., 2018).

But DL offers a unique contribution to neuroscience that goes beyond its role in other fields, in that neural networks can be viewed as models of the brain. The success of DL in matching or surpassing human performance means it is now possible to construct models of circuits that may underlie human intelligence. As a recent review wrote, “Many researchers are excited by the possibility that deep 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” (Saxe et al., 2021).

Broadly, the essential claims 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 they reveal the optimization problem of the brain, or 2) That these models, when trained on sensibly motivated optimization problems, should make novel predictions about the brain’s representations and emergent behavior.

However, given the nascent nature of such approaches and the excitement accompanying the claims, we should examine them carefully. In deep learning and deep reinforcement learning, performance improvements attributed to novel objective functions and algorithms have been shown to instead stem from seemingly minor and often-unstated implementation choices (Tucker et al., 2018; Henderson et al., 2019; Engstrom et al., 2020; Ilyas et al., 2020). Similar criticism has been raised in the context of Neuroscience (Hosseini et al., 2020; Schulz et al., 2020; Abrol et al., 2021). In this
To explore these questions, 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 system that displays beautiful and fascinating properties. In 1971, HPC was shown to contain place cells, neurons which fire if and only if the animal is at particular location(s) (O’Keefe & Dostrovsky, 1971). Later, MEC was shown to contain grid cells, neurons which fire if and only if the animal is at the vertices of a hexagonal lattice (Hafting et al., 2005). As a matter of terminology, we refer to cells that are periodically active at the vertices of either a square or hexagonal lattice as lattice cells, and refer to hexagonal lattice cells as grid cells. Over the past 40 years, the hippocampal formation has proved a rich vein for learning about how the brain organizes spatial and episodic memory, for experimentalists and theorists alike (Section 2), with many mysteries remaining. A recent series of papers (Cueva & Wei, 2018; Banino et al., 2018; Sorscher et al., 2019; Whittington et al., 2020; Nayebi et al., 2021) have used DL to present a story that path integration (PI) (i.e., the task of estimating one’s absolute spatial position in an environment by integrating one’s velocity estimates) drives the formation of grid cells.

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

1. Networks trained on a path integration task almost always learn to optimally encode position, but almost never learn lattice cells (hexagonal or square) to do so.

2. The emergence of lattice cells depends wholly on a specifically chosen encoding of the supervised target, not on the task itself.

3. The grid periods and period ratios likewise depend on hyperparameters choices and are not set by the task.

4. The chosen encoding requires many other hyperparameter choices to produce lattice cells that are highly sensitive, in that small alterations result in loss of lattice cells.

Deep learning produces grid cells and their attendant properties only after making many specific design choices and searching hyperparameter space to obtain such representations, baking grid cells into the task-trained networks.

Our main message is that it is highly improbable that DL models of path integration would have produced grid cells as a novel prediction simply from task-training, had grid cells not already been known to exist. Moreover, it is unclear what added interpretability or understanding these models contribute, beyond or even up to what has already been shown for these particular circuits. 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.

We emphasize that our work would not have been possible without the open-source code released by previous publications, for which the authors should be commended; by making their code available, we have been able to present novel insights that we hope will contribute to a clearer understanding of the risks and rewards of using and interpreting DL models in Neuroscience.

2. Background and related work

Grid cells (Hafting et al., 2005) are specialized neurons found in the medial entorhinal cortex (MEC) of mammals that are tuned to represent the spatial location of the animal as it freely 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 circuit. Here we list some of the properties most relevant to testing grid cell models: 1) The spatial period of the periodic grid response...
is independent of the spatial environment’s size and shape for familiar spaces. 2) The response is updated even in the dark, in the absence of external cues. 3) Grid cells form discrete modules (clusters), such that all cells within a module share a common period and orientation, with monotonically increasing grid scales along the dorso-ventral axis of the MEC (Stensola et al., 2012). Adjacent modules exhibit specific period ratios whose values are all close to 1.4. 4) The grid cells within each module exhibit stable cell-cell relationships across environments and during sleep (Yoon et al., 2013; Trettel et al., 2019; Gardner et al., 2019a; 2022), unlike place cells, which remap across environments and do not exhibit preserved structure during sleep.

The mechanism underlying grid cells is believed to be described by continuous attractor models (Burak & Fiete, 2009b; Fuhs & Touretzky, 2006; Burak & Fiete, 2006). They operate according to an elegant principle: Translation-invariant lateral connectivity within the grid cell network results in Turing instability and pattern formation. These models explain how velocity inputs can be converted into updated spatial estimates by grid cells and make several predictions that have been confirmed in experiments: 1) Stable cell-cell relationships regardless of environment (Yoon et al., 2013; Trettel et al., 2019; Gardner et al., 2019a; 2022); 2) A toroidal attractor manifold in neural state space across awake and sleep states (Gardner et al., 2022; 2019b).

The counterintuitive encoding of position by grid cells (a local, non-periodic variable such as location encoded by a non-local, periodic code) has led to theoretical examination of the properties of the population of grid cells (oftentimes called the grid code), with results showing that the grid code provides three properties not shared with many conventional (unimodal or grandmother cell-like) neural codes: 1) Equi-norm translation-invariant representations for path integration; 2) Exponentially large capacity of internally generated states as a function of neuron number (Fiete et al., 2008; Sreenivasan & Fiete, 2011; Mathis et al., 2012b); 3) Intrinsic error correcting code (Sreenivasan & Fiete, 2011).

3. Experimental Setup

Path integration (PI) is the task of using self-velocity estimates to track one’s spatial position over time, a crucial component of spatial navigation. The central message of DL models of grid cells is that training recurrent networks to path integrate causes the networks to learn lattice cells (Cueva & Wei, 2018; Banino et al., 2018; Sorscher et al., 2019; Whittington et al., 2020; Nayebi et al., 2021).

Many of the previous papers use the following experimental setup. A 2-D bounded spatial environment is created, often times a 2.2 m x 2.2 m open arena. Then, spatial trajectories (i.e. sequences of positions and velocities) are sampled. Networks receive as inputs the initial position and the sequence of velocities from a trajectory, and are trained to output the sequence of positions (Fig. 2) in a supervised manner. However, there are multiple ways of encoding the position, and as we shall show, this choice is critical. Two simple encodings of position are Cartesian (Cueva & Wei, 2018) or polar. Another common encoding scheme is via place cells (PCs), which works by sampling a large population of PCs with positions uniformly distributed in the environment, and then computing each place cell’s activity at each position based on a particular function; two common functions are a Gaussian (Banino et al., 2018; Sorscher et al., 2019) and a Difference of Gaussians (Sorscher et al., 2019; Nayebi et al., 2021). If a PC encoding is used, the networks are trained to predict the high dimensional vector of PC activities. For all encodings, supervised learning is used to train the network via backpropagation through time.

Ratemaps are the primary method used to compare the hidden units of artificial networks to grid cells. They are computed using the following procedure: A trained network is evaluated on longer trajectories that roughly uniformly cover the 2-dimensional environment. As the network integrates instantaneous velocity inputs and outputs the positional encoding (typically a place cell code), each hidden unit’s activity is binned against the true position in the 2-dimensional environment. For details, see Appendix B.

4. Networks trained on path integration tasks learn to optimally encode position, but rarely learn lattice (hexagonal or square) cells

As Sorscher et al. (2019) 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?” We shall demonstrate, in contrast, that most
networks do not converge to a grid-like solution, instead requiring very specific hyperparameter choices, and what governs the structure (when it emerges) is determined entirely by what the programmer bakes in.

We ran large-scale hyperparameter sweeps across common implementation choices: 1) Network Architectures: RNN (Elman, 1990); LSTM (Hochreiter & Schmidhuber, 1997); GRU (Chung et al., 2014); UGRNN (Collins et al., 2017) 2) Activation: Linear; Sigmoid; Tanh; Rectified Linear 3) Optimizers: SGD, Adam (Kingma & Ba, 2017); RMSProp (Hinton et al.) 4) Supervised Targets: Cartesian spatial position; high-dimensional Place Cell (PC) population code with Gaussian tuning curves (Banino et al., 2018); high-dimensional PC population code with Difference-of-Gaussians (DoG) tuning curves (Sorscher et al., 2019; Nayebi et al., 2021) 5) Loss: mean squared error (MSE) on the agent’s Cartesian spatial position (Kanitscheider & Fiete, 2016; Cueva & Wei, 2018); geodesic distance on the agent’s polar spatial position (Kanitscheider & Fiete, 2017a); softmax cross entropy on a high-dimensional population of place cell (PC) units (Banino et al., 2018; Sorscher et al., 2019; Nayebi et al., 2021) 6) Miscellaneous: recurrent dropout, readout dropout, weight regularization, randomization seed.

When training networks on supervised place cell (PC) targets, we additionally swept: 1) The place cells’ receptive field $\sigma$ i.e. the standard deviation of the Gaussian tuning curve (often denoted $\sigma_E \equiv \sigma$ in the literature); 2) Whether the PC population’s fields have homogeneous or heterogeneous receptive fields; 3) In the case of PC targets with DoG tuning curves, the place cell’s surround scale $s$ i.e. the ratio between the inhibitory and excitatory Gaussian’s standard deviations (often denoted $s \equiv \sigma_I/\sigma_E$ in the literature); 4) The number of fields per place cell.

Evaluating the entire hyperparameter volume is computationally prohibitive, so we evaluated a subvolume we considered most consistent with previous approaches as well as most illustrative of previous approaches’ limitations; complete details of every run sweep configurations are provided in Appendix C.

To evaluate whether a network learns to optimally encode its position, we measured the network’s position decoding error using the same methods as prior works (Sorscher et al., 2019; Nayebi et al., 2021). Specifically, we computed position decoding error in a 2.2 m x 2.2 m environment using the networks’ output Cartesian positions (if trained on Cartesian position targets) or using the networks’ decoded positions based on their predicted PC population activity (if trained on PC targets). Any network was considered to have achieved optimal position encoding if its position decoding error fell below 6 cm; this threshold was chosen based on noise inherent in the position decoding algorithm.

In total, we trained 415 networks and found that almost every hyperparameter configuration succeeds in learning to path integrate (87.2%; Fig. 3a), but only few learn lattice cells at all (14.6% of the 87.2%, or 12.7% overall) representations (Fig. 3b). This is consistent with earlier work (Kanitscheider & Fiete, 2016; 2017a) 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 lattice cells emerging as a solution.

5. Lattice cell emergence requires a highly specific choice of supervised target encoding

We next sought to characterize when lattice cells are learnt if the target output encodings are changed. We tested three
different encodings of position in 2d space: a) Cartesian readouts b) Gaussian readouts c) a very specifically selected Difference-of-Gaussian (DoG) readout. We found that (cf. Fig. 4) grid cells do not emerge from Cartesian and Gaussian readout encodings, consistent with an earlier study (Kantscheider & Fiete, 2017b). Only by making the positional readout encoded by a DoG shape of tuning curve sometimes resulted in lattices (Fig. 4ab).

5.1. Grid periods are parameter-dependent and multiple modules do not usually 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 $\frac{1}{4}$ between adjacent scales (Stensola et al., 2012). 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 DoGs, and swept over different scales of the place cell DoG. We found that almost all runs had a unimodal distribution of grid periods (Fig. 5a), meaning the networks learnt only one module of grid cells.

Further, we found that the period of the formed grid-like representation is completely determined by the scale of the externally imposed readout DoG (Fig. 5). The period the grid-like responses in every run increased monotonically with the width of the DoG readout (Fig. 5b). 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 DoG parameter more continuously to compute grid period ratios. These period ratios from adjacent peaks led to non-biological values (Fig. 5c).

5.2. Fourier analysis of Turing instability explains the preceding empirical results

Why do only Difference-of-Gaussians (DoG) place cell targets produce lattice cells? The reason is the same as for pattern-forming models of grid cells which showed that a difference of Gaussians recurrent interaction is sufficient to generate periodic activity patterns (Burak & Fiete, 2009b) in recurrent network models of grid cells, and can be understood through Fourier analyses (Burak & Fiete, 2009b; Khona et al., 2021). We restate the essence of the analyses here, and explain why it explains the current results. In an RNN with dynamics $\dot{r}(x) = -r(x) + g(W \ast r)$, where $x$ designates the neural index (in a continuum approximation for neurons), $W \ast r$ designates the total (integrated) inputs from the network at the neuron at location $x$, and $g$ is the neural non-linearity, let the recurrent weight interactions $W$ be given by:

$$f(\Delta x) = \alpha_E \exp\left(-\frac{(\Delta x)^2}{2\sigma_E^2}\right) - \alpha_I \exp\left(-\frac{(\Delta x)^2}{2\sigma_I^2}\right)$$

where $\Delta x$ refers to the difference of indices between the neural pair linked by the weights. The Fourier transform of
This interaction is given by

$$\tilde{f}(k) = \alpha_E \sigma_E \exp \left( -\frac{\sigma_E^2 k^2}{2} \right) - \alpha_I \sigma_I \exp \left( -\frac{\sigma_I^2 k^2}{2} \right).$$  \hspace{1cm} (2)

Here $\alpha_E$ ($\alpha_I$) denotes the strength and $\sigma_E$ ($\sigma_I$) denotes the scale of excitation (inhibition). For linearized dynamics that approximate $\tilde{f}(x) \sim -r(x) + f(\Delta x) * r$ (i.e. $g$ has been linearized), the solution will be periodic if the maxima of $\tilde{f}(k)$, given by $k^* = \pm 2 \sigma_E^2 \sigma_I^{-1} \log \left( \frac{\alpha_E \sigma_E^2}{\alpha_I \sigma_I^2} \right)$, contains sufficient power and if $k^* \neq 0$. Specifically, the condition for pattern formation is $\tilde{f}(k^*) > 1$ (Burak & Fiete, 2009a; Khona et al., 2021). In particular, the inhibitory surround contained in $f(\Delta x)$, with strength $\sigma_I$, is key to pattern 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 theory in Sorscher et al. (2020); Dordek et al. (2016) similarly shows that the Fourier transform of the desired readout unit activity (in particular, its second-moment matrix) sets the activity of the RNN units. This readout matrix takes on the role of the recurrent interaction matrix of the continuous attractor grid cell model described above (Burak & Fiete, 2009b). The conditions for a pattern forming solution in supervised training-based RNN setup for obtaining grid cells are therefore the same as described above. That is, for Gaussian readout functions ($\alpha_I = 0$), the peak of their Fourier spectrum is at the origin, causing DC modes to dominate and no latencies to form regardless of the width $\sigma_E$. Thus, Gaussian readout positional encodings are predicted to not produce grid cells. Through a similar analysis, Cartesian readouts will also not produce grid cells. And only specifically tuned DoG readouts produce grid cells under the architectures considered here and in Sorscher et al. (2020). Our large-scale hyperparameter sweeps confirm this.

5.3. The importance of unexplored implementation details

A seemingly minor implementation detail that is not mentioned in the main texts or supplements (to the best of our knowledge) in several of the preceding papers (Sorscher et al., 2019; Nayebi et al., 2021; Sorscher et al., 2020) also proves critical to the emergence of grid cells. These papers refer to a Difference-of-Gaussian (DoG) supervised target function (Eqn. 1), but we discovered their code uses something different: a Difference-of-Softmaxes (DoS). Specifically, the tuning curves are given by:

$$\tilde{f}(\Delta x_p) = \text{Softmax} \left( -\frac{(\Delta x_p)^2}{2\sigma_E^2} \right) - \text{Softmax} \left( -\frac{(\Delta x_p)^2}{2\sigma_I^2} \right)$$  \hspace{1cm} (3)

where the softmaxes are taken over the place cell population (indexed here by $p$). This leads to an effective non-uniform averaging of the outputs. Everything previously labeled DoG should actually be labeled DoS; we keep the authors’ terminology in the previous sections, but change the terminology to DoS in Fig. 7bc to distinguish the previous “DoG” (actually DoS) from the true DoG.

To understand why, recall from Section 5.2 that for lattice cells to emerge, Fourier power (i.e. the eigenvalues of the place cell second-moment matrix $\Sigma = P^T P$) must peak on an annulus of a sufficiently large radius; if the radius is too small, lattice cells will not emerge. With DoS instead of DoG, the maximal eigenvalues of the second-moment matrix fall on an annulus with radius 3 (Fig. 7a), which is clearly separated from the origin, giving rise to lattice cells. But a DoG with the same parameters would produce an annulus radius of 1 (Fig. 7c), too small for lattice cells to emerge. We trained ideal grid-forming ReLU RNN networks on supervised PC targets with true DoG tuning curves, sweeping the receptive field $\sigma$ and surround scale $s$. We found that using true Difference-of-Gaussian tuning curves did not result in grid cells (Fig. 7b).
In sum, the simple objective of only asking for accurate spatial representation through path integration is not sufficient for obtaining grid cells, and requires very specific additional choices.

6. Grid cells disappear with heterogeneously tuned readouts units

Experimentally recorded place cells differ significantly from overly simplified single-scale, single-field homogeneous Gaussian functions (and as we have noted above, are even further from being DoG functions). Many place cells have multiple fields (Rich et al., 2014; Eliav et al., 2021; Souza et al., 2018). This naturally leads to the question: Will readout target functions with heterogeneous scales or multiple fields per place cell still produce grid cells?

Because the Fourier analysis in the above argument is applicable when the population of place cells encodes the readout in an approximately translationally-invariant manner (this happens when the place cell population is distributed isotropically in the room, and every place cell has a single peak of the same scale), we cannot predict the outcome of heterogeneous readouts.

To maximally favor the chances of the ANN approach to lead to the formation of lattice responses, we allow as before for the readouts to have DoS profiles. We tested what effect heterogeneous DoS scales have on the formation of grid cells. We trained ideal grid-forming networks (RNN, ReLU, $\sigma = 0.12$ cm, $s = 2.0$, 4 seeds) on heterogeneous DoS PC supervised targets with receptive field $\sigma \sim \text{Uniform}(0.06, 0.18)$ cm and surround scale $s \sim \text{Uniform}(1.5, 2.5)$; these distributions were chosen to ensure the expected values were ideal for grid cell formation. We found heterogeneity in the PC scales prevents the formation of grid cells (Fig. 6 left). Specifically, heterogeneity causes a clear decrease in the bulk of the networks’ grid scores, as well as a decrease in the high grid score tail. Of the few remaining units (13 / 2048) with high grid scores, the distribution of grid periods remained unimodal (Fig. 6 right), showing that grids of multiple scales are not learnt. The networks trained on heterogeneous PC targets achieved equivalent position decoding error as networks trained on homogeneous PC targets, further demonstrating that optimal position encoding does not require lattice representations.

7. Why ANNs that path integrate achieve high predictivity of MEC data

We conclude by introducing a puzzle. A recent NeurIPS 2021 spotlight (Nayebi et al., 2021) claimed that networks trained on single-field single-scale Difference-of-Softmax targets explain variance in mouse MEC neural activity at nearly 100% of the level of variance explained by MEC activity from other mice. In contrast, our results demonstrate that these networks learn very few lattice cells and produce only unimodally distributed grid period distributions, and require artificial place cells inconsistent with biological place cells to do so. Consequently, a question must be answered: how are these networks able to predict mouse MEC neural activity so well?

The data preprocessing and analysis code is not public, so we are unable to investigate this question in detail. However, we offer a conjecture with preliminary supporting evidence. The analysis of Nayebi et al. (2021) 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 conjectured that different combinations of architectures and activations achieve different neural predictivity scores because different architecture-activation combinations learn dynamical systems of different intrinsic dimensionalities that then provide richer or poorer bases for linear regressions.

To explore our conjecture, we trained the networks considered in Nayebi et al. (2021): 4 architectures (RNN, LSTM, GRU, UGRNN) and 4 activations (Linear, Sigmoid, Tanh, ReLU), adding 5 random seeds (0 through 4) per architecture-activation pair. For each trained network, we computed a standard linear measure of the intrinsic dimensionality (called participation ratio (Litwin-Kumar et al., 2017)) of the network’s rate maps, and then plotted each network’s participation ratio against the published neural predictivity score. We chose a linear measure of the networks’ intrinsic dimensionalities since linear regression cannot fit nonlinear patterns in the data. We found a clear trend that networks with higher (lower) dimensional rate maps have higher (lower) neural predictivity scores (Fig. 8).

We caution that this correlation between network dimen-
Figure 7. Other changes to Difference-of-Gaussians affect the formation of grid cells: (a) The existence of grid solutions even with DoG readout encodings is highly sensitive to parameters such as the target function receptive field $\sigma$, 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 true DoG shows that the DoG with normalization, as used by Sorscher et al. (2019); Nayebi et al. (2021); Sorscher et al. (2020), is critical for grid cell formation, (right) Ratemaps of the highest scoring cells from the true DoG networks. (c) Computing the Fourier transform (right of each pair) of the place cell second-moment matrix (left of each pair) explains why the particular choice of normalization affects representations.

Figure 8. Networks with higher (lower) dimensional rate maps display higher (lower) “neural predictivity” of mouse MEC rate maps. This suggests that linear regression-based comparisons of artificial and biological networks may find “better” networks simply because those networks provide higher dimensional bases for the regressions. Each dot is an architecture-activation pair, with participation ratio (Litwin-Kumar et al., 2017) averaged over 5 runs. Neural predictivity scores from Fig. 2A of Nayebi et al. (2021).

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 minima (Fig. 9 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. 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. 9 bottom). In other words, there is typically a one-to-many mapping widely used and cited in vision, language and audition, may also lead to artefactual conclusions. DL-based models may not actually be close models of biological neural circuits, but rather may merely provide higher-dimensional bases than alternative models and thus trivially achieve higher correlation scores.
Figure 9. **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.

From a loss function to its set of solutions.

To break this degeneracy of multiple minima and arrive at truly predictive models or a better understanding of the brain’s optimization problems, we must acknowledge, understand, and model the host of specific implicit biases and constraints present in the biological system we are trying to model. It is untenable to expect and claim success without doing so. This is what we refer to as an informal neural ‘no-free-lunch’.

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 robust enough and abstracted enough to be predictive of population dynamics in a neural circuit without the addition of detailed biological constraints. This can explain the success of the population-level analyses of the visual pathway (Kell et al., 2018; Bashivan et al., 2019), as well as the population-level low-dimensional dynamics of circuits solving inference tasks (Sussillo & Barak, 2013; Schaeffer et al., 2020; Voigts et al., 2022). In these cases, the emergent low-dimensional solutions are fundamental features of any system that solve the task, and by construction need not be specific to brains.

Coming back to grid cells, we have conclusively shown that they do not generically arise in networks trained to path-integrate.

This raises the question of what different additional architectural, hyperparameter, and constraint choices had to be made in previous papers (Banino et al., 2018; Sorscher et al., 2019; 2020) to obtain grid-like tuning, given that they used a path integration objective with Gaussian place cell targets. Indeed, we argue that the question of scientific interest is to explore and carefully characterize the conditions under which a particular tuning does and does not emerge.

To this end, the question is what other ingredient(s) should be added to the task of path integration to always and robustly obtain grid cells? Theoretical work on grid cell representations (Fiete et al., 2008; Sreenivasan & Fiete, 2011; Mathis et al., 2012b) suggests that the following two factors are important features of the grid cell code: 1) a very large coding range and 2) the related property of robustness/intrinsic error correcting coding. Adding these properties to the loss are likely to be a more principled way to obtain grid-like fields rather than by hand-designing a specific and not biologically motivated DoG readout. Consistent with this, the very large amount of dropout required in Banino et al. (2018); Cueva & Wei (2018) suggests that the coding-theoretic insights on intrinsic error-correction properties of grid cells and their related large capacity may indeed be key ingredients to produce grid cells. There are a number of specific properties of the grid cell code elucidated by theoretical arguments outlined before, which we hypothesize form a sufficient set of biologically and computationally relevant properties 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 (Fuhs & Touretzky, 2006; Burak & Fiete, 2009b); 4) a high representational capacity (Burak & Fiete, 2008; Sreenivasan & Fiete, 2011; Mathis et al., 2012a); 5) intrinsic error correcting capabilities (Burak & Fiete, 2008; Sreenivasan & Fiete, 2011); and finally, 6) uniformly distributed and low spatial information per cell. In other words, the total spatial information of the grid code should be equally distributed across all modules and cells, and this distribution should be roughly equal.
References


Nayebi, A., Attinger, A., Campbell, M. G., Hardcastle, K., Low, I. I., Mallory, C. S., Mel, G. C.,


A. Selecting the Grid Score Threshold

What qualifies as a grid cell? The most commonly used method of quantifying grid cells is via 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. Experimentalists have used thresholds of 0.3 (Solstad et al., 2008) and 0.349 (Campbell et al., 2018) on biological neurons, whereas computationalists have used 0.3 (Nayebi et al., 2021) and 0.37 (Banino et al., 2018; Sorscher et al., 2019) on artificial neurons. For artificial neurons, we believe that these thresholds are far too low. This is because biological neurons are noisy and undersampled, whereas artificial neurons are noisless and oversampled, and the grid score decreases with missing/noisy rate maps. Consequently, as shown in Fig. 10, it is common for artificial neurons to achieve grid scores above 0.35 without being grid-like. Even artificial neurons with grid scores above 0.8 are arguably not grid cells. After internal disagreement, we compromised at a grid score threshold of 0.8; one author would like to note that they believe artificial neurons should be held to the highest standard possible and argued for a threshold of 1.15.

B. 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 publications was challenging. The original experimental work used 5 cm by 5 cm bins (Hafting et al., 2005) in a 2.2 m by 2.2 m arena, meaning the number of bins should be 44 x 44. However, the deep learning papers (Banino et al., 2018; Sorscher et al., 2019; Nayebi et al., 2021) revealed discrepancies between text and code.

Banino et al. (2018)’s text notes using 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 x 32 square grid spanning each environment"), but their code ¹ used 20 x 20 bins of 11 cm x 11 cm. Sorscher et al. (2019) noted using 2 cm x 2 cm bins ("Grid score was evaluated as in Banino et al. (2018). A spatial ratemap was computed for each neuron by binning the agent’s position into 2 cm x 2 cm bins, and computing the average firing rate within each bin.") but their code ² similarly used 20 x 20 bins.

1https://github.com/deepmind/grid-cells/blob/master/train.py#L201
2https://github.com/ganguli-lab/grid-pattern-formation/blob/master/

Figure 10. Grid scores are an imperfect way to characterize grid cells. Example rate maps from three grid score ranges: [0.35, 0.45], [0.8, 0.9], [1.15, ∞). We considered three grid score thresholds: 0.3 (used by experimentalists on biological data), 0.8 (decent probability of finding grid cells), 1.15 (high probability of finding grid cells). Only the grid cell with score 1.26 looks like a grid cell. Grid scores for each rate map are shown above each rate map.
Nayebi et al. (2021) noted using 5 cm x 5 cm bins (“Nayebi: “We bin the positions in each environment using 5 cm bins, following prior work [Harcastle et al., 2017, Butler et al., 2019, Low et al., 2020]. Thus, the 100cm² environment used 400 (20 x 20) bins, the 150cm² environment used 900 (30 x 30) bins, and the 400cm 1D track used 80 bins.”) but their code similarly used 20 x 20 bins. The code suggests that Nayebi et al. (2021) used the grid scorer of Sorscher et al. (2019), who in turn used the grid scorer of Banino et al. (2018).

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. 11), so we used 44 x 44 bins since this yields bins of size 5 cm x 5 cm, matching the norm in the experimental physiology literature, to which models are compared.

C. Sweep Configurations

C.1. Cartesian Position with Mean Squared Error

```
method: grid
metric:
goal: minimize
name: pos_decoding_err
parameters:
  Ng:
    values:
    - 4096
  Np:
    values:
    - 2
  activation:
    values:
    - 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
```

Figure 11. 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).
C.2. Gaussian Place Cells with Cross Entropy Loss

method: grid
metric:
goal: minimize
name: pos_decoding_err
parameters:
Ng:
  values:
  - 4096
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
  - glorot_normal
  - orthogonal
is_periodic:
  values:
  - false
learning_rate:
  values:
  - 0.0001
n_epochs:
  values:
  - 10
n_grad_steps_per_epoch:
  values:
  - 10000
n_place_fields_per_cell:
  values:
  - 1
optimizer:
  values:
  - adam
No Free Lunch from Deep Learning in Neuroscience: A Case Study through Models of the MEC-HPC Circuit

place_cell_rf:
values:
- 0.12
- 0.2
place_field_loss:
values:
- crossentropy
place_field_normalization:
values:
- global
place_field_values:
values:
- gaussian
readout_dropout:
values:
- 0
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

C.3. True Difference of Gaussian Place Cells with Cross Entropy Loss

method: grid
metric:
goal: minimize
name: pos_decoding_err
parameters:
Ng:
values:
- 4096
Np:
values:
- 512
activation:
values:

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:
- 0.1
- 0.12
- 0.2
- 0.3
place_field_loss:
values:
- crossentropy
place_field_normalization:
values:
- global
place_field_values:
values:
- true_difference_of_gaussians
readout_dropout:
values:
- 0
rnn_type:
values:
  - RNN
seed:
  values:
  - 0
  - 1
  - 2
sequence_length:
  values:
  - 20
surround_scale:
  values:
  - 1.5
  - 2
  - 2.5
weight_decay:
  values:
  - 0.0001

C.4. “Difference of Gaussian” (Difference of Softmax)
Place Cells with Cross Entropy Loss, Sweeping
Receptive Field

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
optimizer:
  values:
  - adam
place_cell_rf:
  values:
  - 0.04
  - 0.05
  - 0.06
  - 0.07
  - 0.08
  - 0.09
place_field_loss:
  values:
  - crossentropy
place_field_normalization:
  values:
  - global
place_field_values:
  values:
  - difference_of_gaussians
readout_dropout:
  values:
  - 0
rnn_type:
  values:
  - RNN
seed:
  values:
  - 0
  - 1
  - 2
sequence_length:
  values:
surround_scale: values:
  - 20

weight_decay: values:
  - 2

C.5. “Difference of Gaussian” (Difference of Softmax)
Place Cells with Cross Entropy Loss, Sweeping
Surround Scale

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:
    - 10
  n_grad_steps_per_epoch:
    values:
    - 10000
  n_place_fields_per_cell: values:
    - 1
  optimizer:
    values:
    - adam
  place_cell_rf:
    values:
    - 0.12
    - 0.2
  place_field_loss:
    values:
    - crossentropy
  place_field_normalization:
    values:
    - global
  place_field_values:
    values:
    - difference_of_gaussians
  readout_dropout:
    values:
    - 0
  rnn_type:
    values:
    - RNN
  seed:
    values:
    - 0
    - 1
  sequence_length:
    values:
    - 20
  surround_scale:
    values:
    - 1.5
    - 1.75
    - 2
    - 2.25
    - 2.5
    - 3
  weight_decay:
    values:
    - 0.0001

C.6. “Difference of Gaussian” (Difference of Softmax)
Place Cells with Cross Entropy Loss,
Heterogeneous Scales

method: grid
metric:
  goal: minimize
  name: pos_decoding_err
parameters:
  Ng:
    values:
No Free Lunch from Deep Learning in Neuroscience: A Case Study through Models of the MEC-HPC Circuit

- 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
- Uniform( 0.10 , 0.14 )
- Uniform( 0.08 , 0.16 )
- Uniform( 0.06 , 0.18 )
place_field_loss:
values:
- crossentropy
place_field_normalization:
values:
- global
place_field_values:
values:
- difference_of_gaussians
readout_dropout:
values:
- 0
rnn_type:
values:
- RNN
seed:
values:
- 0
- 1
- 2
- 4
sequence_length:
values:
- 20
surround_scale:
values:
- 2
- Uniform( 1.90 , 2.10 )
- Uniform( 1.75 , 2.25 )
- Uniform( 1.50 , 2.50 )
weight_decay:
values:
- 0.0001

C.7. Softmax of Difference of Squared Distances Place Cells with Cross Entropy Loss

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:
initializer:
values:
- glorot_uniform

is_periodic:
values:
- false

learning_rate:
values:
- 0.0001

n_epochs:
values:
- 10

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
- 0.2

place_field_loss:
values:
- crossentropy

place_field_normalization:
values:
- global

place_field_values:
values:
- softmax_of_differences

readout_dropout:
values:
- 0

rnn_type:
values:
- RNN

seed:
values:
- 0
- 1
- 2

sequence_length:
values:
- 20

surround_scale:
values:
- 1.5
- 2

weight_decay:
values:
- 0.0001

weight_decay:
values:
- 0.0001

weight_decay:
values:
- 0.0001