RELATING TRANSFORMERS TO MODELS AND NEURAL REPRESENTATIONS OF THE HIPPOCAMPAL FORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Many deep neural network architectures loosely based on brain networks have recently been shown to replicate neural firing patterns observed in the brain. One of the most exciting and promising novel architectures, the Transformer neural network, was developed without the brain in mind. In this work, we show that transformers, when equipped with recurrent position encodings, replicate the precisely tuned spatial representations of the hippocampal formation; most notably place and grid cells. Furthermore, we show that this result is no surprise since it is closely related to current hippocampal models from neuroscience. We additionally show the transformer version offers dramatic performance gains over the neuroscience version. This work continues to bind computations of artificial and brain networks, offers a novel understanding of the hippocampal-cortical interaction, and suggests how wider cortical areas may perform complex tasks beyond current neuroscience models such as language comprehension.

1 INTRODUCTION

The last ten years have seen dramatic developments using deep neural networks, from computer vision (Krizhevsky et al., 2012) to natural language processing and beyond (Vaswani et al., 2017). During the same time, neuroscientists have used these tools to build models of the brain that explain neural recordings at a precision not seen before (Yamins et al., 2014; Banino et al., 2018; Whittington et al., 2020). For example, representations from convolutional neural networks (Lecun et al., 1998) predict neurons in visual and inferior temporal cortex (Yamins et al., 2014; Khaligh-Razavi & Kriegeskorte, 2014), representations from transformer neural networks (Vaswani et al., 2017) predict brain representations in language areas (Schrimpf et al., 2020), and lastly recurrent neural networks (Cueva & Wei, 2018; Banino et al., 2018; Sorscher et al., 2019) have been shown to recapitulate grid cells (Hafting et al., 2005) from medial entorhinal cortex. Being able to use models from machine learning to predict brain representations provides a deeper understanding into the mechanistic computations of the respective brain areas, and offers deeper insight into the nature of the models.

As well as using off-the-shelf machine learning models, neuroscience has developed bespoke deep learning models (mixing together recurrent networks with memory networks) that learn neural representations that mimic the exquisite *spatial* representations found in hippocampus and entorhinal cortex (Whittington et al., 2020; Uria et al., 2020), including grid cells (Hafting et al., 2005), band cells (Krupic et al., 2012), and place cells (O'Keefe & Dostrovsky, 1971). However, since these models are bespoke, it is not clear whether they, and by implication the hippocampal architecture, are capable of the general purpose computations of the kind studied in machine learning.

In this work we 1) show that transformers (with a little twist) recapitulate spatial representations found in the brain; 2) show a close mathematical relationship of this transformer to current hippocampal models from neuroscience (with a focus on Whittington et al. (2020) though the same is true for Uria et al. (2020)); 3) offer a novel take on the computational role of the hippocampus, and an instantiation of hippocampal indexing theory (Teyler & Rudy, 2007); 4) offer novel insights on the role of positional encodings in transformers. 5) discuss whether similar computational principles might apply to broader cognitive domains, such as language, either in the hippocampal formation or in neocortical circuits.

Note, we are not saying the brain is closely related to transformers because it learns the same neural representations, instead we are saying the relationship is close because we have shown a mathe-

matical relationship between transformers and carefully formulated neuroscience models of the hippocampal formation. This relationship helps us get a better understanding of hippocampal models, it also suggests a new mechanism for place cells that would not be possible without this mathematical relationship, and finally it tells us something formal about position encodings in transformers.

2 TRANSFORMERS

Transformer Neural Networks (Vaswani et al., 2017) are highly successful machine learning algorithms. Originally developed for language, transformers perform well on other tasks that can be posed sequentially, such as mathematical understanding, logic problems (Brown et al., 2020), and image processing (Dosovitskiy et al., 2020).

Transformers accept a set of observations; $\mathbb{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_T\}$ ($\boldsymbol{x}_t$ could be a word embedding or image patch etc), and aim to predict missing elements of that set. The missing elements could be in the future, i.e. $\boldsymbol{x}_{t>T}$, or could be a missing part of a sentence or image, i.e. $\{\boldsymbol{x}_1 = \texttt{the}, \boldsymbol{x}_2 = \texttt{cat}, \boldsymbol{x}_3 = \texttt{sat}, \boldsymbol{x}_4 = \texttt{?}, \boldsymbol{x}_5 = \texttt{the}, \boldsymbol{x}_6 = \texttt{mat}\}$.

Self-attention. The core mechanism of transformers is self-attention. Self-attention allows each element to 'attend' to all other elements, and update itself accordingly. In the example data-set above, the 4th element (`?`) could attend to the 2nd (`cat`), 3rd (`sat`), and 6th (`mat`) to understand it should be `on`. Formally, to attend to another element each element ($\boldsymbol{x}_t$ is a row vector) emits a query ($\boldsymbol{q}_t = \boldsymbol{x}_t\boldsymbol{W}_q$) and compares it to other elements keys ($\boldsymbol{k}_\tau = \boldsymbol{x}_t\boldsymbol{W}_k$). Each element is then updated using $\boldsymbol{y}_t = \sum_\tau \kappa(\boldsymbol{q}_t, \boldsymbol{k}_\tau)\boldsymbol{v}_\tau$, where $\kappa(\boldsymbol{q}_t, \boldsymbol{k}_\tau)$ is kernel describing the similarity of $\boldsymbol{q}_t$ to $\boldsymbol{k}_\tau$ and $\boldsymbol{v}_\tau$ is the value computed by each element $\boldsymbol{v}_\tau = \boldsymbol{x}_t\boldsymbol{W}_v$. Intuitively, the similarity measure $\kappa(\boldsymbol{q}_t, \boldsymbol{k}_\tau)$ places more emphasis on the elements that are relevant for prediction; in this example, the keys may contain information about whether the word is a noun, verb or adjective, while the query may 'ask' for any elements that are nouns or verbs - elements that match this criteria (large $\kappa(\boldsymbol{q}_t, \boldsymbol{k}_\tau)$, i.e. `cat, sat, mat`) are 'attended' to and therefore contribute more to the output y_t. Typically, the similarity measure is a softmax i.e. $\kappa(\boldsymbol{q}_t, \boldsymbol{k}_\tau) = \frac{e^{\beta \boldsymbol{q}_t \cdot \boldsymbol{k}_\tau}}{\sum_{\tau'} e^{\beta \boldsymbol{q}_t \cdot \boldsymbol{k}_{\tau'}}}$.

These equations can be succinctly expressed in matrix form, with all elements updated simultaneously:

$$\boldsymbol{y}_t = softmax(\frac{\boldsymbol{q}_t\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \qquad \rightarrow \qquad \boldsymbol{Y} = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{1}$$

Here $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ are matrices with rows filled by $\boldsymbol{q}_t$, $\boldsymbol{k}_t$, $\boldsymbol{v}_t$ respectively, and the softmax is taken independently for each row. After this update, each $\boldsymbol{y}_t$ is then sent through a deep network ($f_\theta(\cdots)$) typically consisting of residual (He et al., 2016) and layer-normalisation (Ba et al., 2016b) layers to produce $\boldsymbol{z}_t = f_\theta(\boldsymbol{y}_t)$. $\boldsymbol{Z}$ is the output of the transformer which can then be used for prediction, or sent through subsequent transformer blocks.

Position encodings. Self-attention is permutation invariant and so tells you nothing about order of the inputs. Should the data be sequential (i.e. meaning depends on the order of elements, such as in language, or navigation as we will see later!), it is necessary to additionally encode the position/ where $\boldsymbol{x}$ is in the sequence. This is typically done by adding a 'position encoding' that uniquely identifies each time-step ($\boldsymbol{e}_t$ - typically sines and cosines) to each input: $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_t + \boldsymbol{e}_t$. Alternatively the position embedding can be appended i.e. $\boldsymbol{h}_t = [\boldsymbol{x}_t, \boldsymbol{e}_t]$, with self attention then performed using $\boldsymbol{h}_t$ as input.

3 TRANSFORMERS LEARN ENTORHINAL REPRESENTATIONS

Here we show that transformers (with a small modification) recapitulate spatial representations - grid and band cells - when trained on tasks that require abstract spatial knowledge.

Spatial understanding task. The task (more detail in Appendix) is to predict upcoming sensory observations $\boldsymbol{x}_{t+1}$ conditioned on taking an action $\boldsymbol{a}_t$ while moving around spatial environments (Figure 1a). For example, after seeing $\{(\boldsymbol{x}_1 = \texttt{cat}, \boldsymbol{a}_1 = \texttt{North}), (\boldsymbol{x}_2 = \texttt{dog}, \boldsymbol{a}_2 = \texttt{East}), (\boldsymbol{x}_3 = \texttt{frog}, \boldsymbol{a}_3 = \texttt{South}), (\boldsymbol{x}_4 = \texttt{pig}, \boldsymbol{a}_4 = \texttt{West}), (\boldsymbol{x}_5 = \texttt{?}, \boldsymbol{a}_5 = \cdots)\}$, the aim is to predict $\boldsymbol{x}_5 = \texttt{cat}$. For simplicity, we treat sensory observations as one-hot vectors, thus the prediction problem is a classification problem.

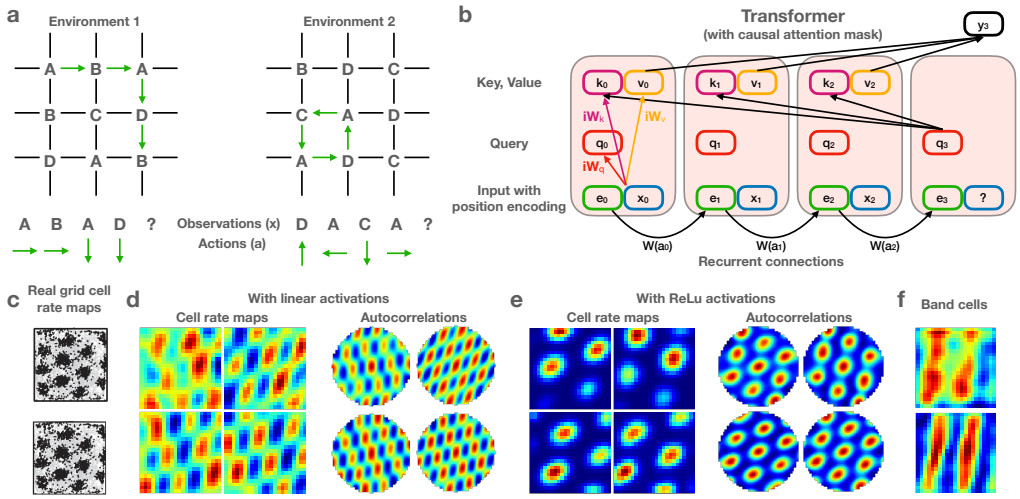

Figure 1: **(a)** Sequence prediction in spatial navigation tasks test abstract spatial understanding since some sensory predictions can only be done by knowing (generalising) certain rules e.g. `North` + `East` + `South` + `West` = 0 or `Parent` + `Sibling` + `Niece` = 0. Note, we use sequences drawn from much larger graphs. **(b)** Transformer with recurrent position encodings. **(c)** Real grid cell rate-maps (Hafting et al., 2005). **(d-f)** Learned position embedding rate-maps (i.e. average activity at each spatial location; plots are spatially smoothed). **(d-e)** Resembling grid cells with **(e)** linear activation or **(e)** ReLu activation post transition. **(f)** Resembling band cells (Krupic et al., 2012).

When faced with an unseen stimulus-action pair (e.g. $\boldsymbol{x}_4 = \texttt{pig}, \boldsymbol{a}_4 = \texttt{West}$ above; an action you have never taken at that stimulus before), successful prediction requires more than just remembering specific sequences of stimulus-action pairs; knowledge of the rules of space must be known; i.e. `North` + `East` + `South` + `West` = 0 allows prediction of $\boldsymbol{x}_5 = \texttt{cat}$. Crucially, such rules *generalise* to any 2D spaces and may therefore be *transferred* to aid prediction in entirely novel 2D environments. This is powerful, since unobserved relations between observed stimuli can be inferred in a zero-shot manner.

However, these relational rules are not 'known' *a priori* and therefore must be learnt. We therefore train across multiple different spatial environments which share the same underlying 4-connected Euclidean structure (Figure 1a) - this means the model must learn and generalise the abstract structure of space to use for prediction in new environments.

To perform on these tasks, the three modifications to the transformer are:

1. Recall equation 1; $\boldsymbol{y}_t = softmax(\frac{\boldsymbol{q}_t \boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}$, where $\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{W}_q$, $\boldsymbol{K} = \boldsymbol{H}\boldsymbol{W}_k$, $\boldsymbol{V} = \boldsymbol{H}\boldsymbol{W}_v$, and $\boldsymbol{H}$ is a matrix of inputs and position encodings (i.e. its rows are $\boldsymbol{h}_t = [\boldsymbol{x}_t, \boldsymbol{e}_t]$). We restrict these weight matrices such that queries ($\boldsymbol{Q}$) and keys ($\boldsymbol{K}$) are the same; $\boldsymbol{Q}, \boldsymbol{K} = \boldsymbol{E}\boldsymbol{W}_e$. We refer to this matrix as $\tilde{\boldsymbol{E}}$. Thus the keys and queries only focus on *position* encodings. Meanwhile, values are exclusively dependent on the *stimulus* component of $\boldsymbol{H}$ i.e. $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_x$. We refer to this matrix as $\tilde{\boldsymbol{X}}$.

$$\boldsymbol{y}_t = softmax(\frac{\boldsymbol{q}_t \boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \qquad \rightarrow \qquad \boldsymbol{y}_t = softmax(\frac{\tilde{\boldsymbol{e}}_t \tilde{\boldsymbol{E}}^T}{\sqrt{d_k}})\tilde{\boldsymbol{X}} \tag{2}$$

   This is an extreme version of the realisation that, in transformers, best performance is when position encodings are used to compute keys and queries, but not values.

2. We use causal transformers; the key and value matrices contain the projected position encodings and sensory stimuli respectively at all *previous* time-steps (i.e. $\boldsymbol{e}_{<t}$ and $\boldsymbol{x}_{<t}$). This is equivalent to causal 'unmasking' as the agent wanders the environment accumulating new experiences (not-yet-experienced stimulus-position pairs are inaccessible to the agent). Meanwhile the query at each time-point is the *present* positional encoding $\boldsymbol{e}_t$.

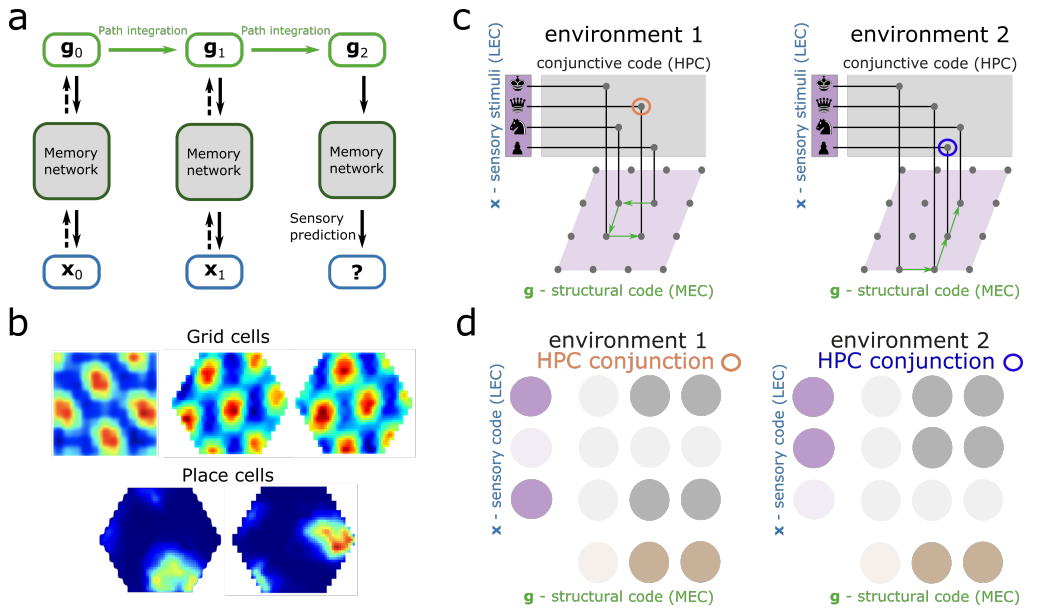

Figure 2: **(a)** The TEM model, with a path integration component (equation 3) and a memory network component (equation 5 and 6). TEM path integrates $\boldsymbol{g}$ and makes sensory predictions $\boldsymbol{x}$ via its memory network (dashed lines are additional connections for inference). **(b)** TEM recapitulates a host of empirically described cell representations (Whittington et al., 2020). Top/bottom row: example TEM MEC/Hippocampal representations (plots are spatially smoothed). Figures adapted from Whittington et al. (2020). **(c)** Schematic of TEM (adapted from Sanders et al. (2020)), showing that the *same* cortical representations (LEC and MEC) are reused in different environments allowing for generalisation, facilitated by *different* hippocampal combinations. **(d)** The TEM hippocampal conjunction is an outer product - cells receive input from particular MEC and LEC cells.

3. The position encodings are recurrently generated (as in Wang et al. (2019); Liu et al. (2020)); $\boldsymbol{e}_{t+1} = \sigma(\boldsymbol{e}_t \boldsymbol{W}_a)$, where $\boldsymbol{W}_a$ is a *learnable* action-dependent weight matrix, and $\sigma(\cdots)$ is a non-linear activation function. This means that unlike traditional transformers, position encodings can be optimised and not the same for every sequence. It now becomes interesting to see what representations are learned.

These modifications are sufficient to learn spatial representations, in the position encodings, that mimic representations observed in the brain (Figure 1C; see Appendix for model and training details). The rest of this paper now explains why this is not a surprising result; namely we show that a transformer with recurrent positional encodings is closely related to current neuroscience models of the hippocampus and surrounding cortex (Whittington et al., 2020; Uria et al., 2020). Here we focus on the Tolman-Eichenbaum Machine (TEM) (Whittington et al., 2020), though the same principles apply for Uria et al. (2020).

The critical points are: 1) the memory component of TEM can be viewed as a transformer self-attention, since the TEM memory network is analogous to a Hopfield network (Hopfield, 1982) which have recently been shown to be closely related to transformers (Ramsauer et al., 2020); 2) TEM path integration (see below) can be viewed as a way to learn a position encoding.

4 TEM

The Tolman-Eichenbaum Machine (TEM; Figure 2, further details in Appendix) is a neuroscience model that captures many known neural phenomena in hippocampus (HPC) and entorhinal cortex (medial/lateral; MEC/LEC). TEM is a sequence learner trained on tasks exactly like the one described in the previous section. TEM consists of two parts;

1) A module that aims to understand where it is in space, using a representation $\boldsymbol{g}$ to represent location. To update its location, TEM uses *path-integration* - the accumulation of self movement vectors $\boldsymbol{a}$ - enacted in a recurrent neural network:

$$\boldsymbol{g}_{t+1} = \sigma(\boldsymbol{g}_t \boldsymbol{W}_a) \tag{3}$$

Where $\boldsymbol{W}_a$ is a learnable action dependent weight matrix and $\sigma(\cdots)$ is a non-linear activation function. It is in this path-integrating representation$\boldsymbol{g}$ that TEM learns grid and other entorhinal cells for self-localisation (Figure 2b).

2) To make sensory predictions, location representations $\boldsymbol{g}$ alone are not enough; they must each link to a sensory observation $\boldsymbol{x}$, corresponding to the stimulus at that position. Note that these links are specific to an environment, since each environment consists of a different arrangement of stimuli in space (i.e. different stimulus-position pairings).

The linking is done by binding every element of $\boldsymbol{g}$ with every element of $\boldsymbol{x}$, in other words an outer product that is flattened back into a vector;

$$\boldsymbol{p} = flatten(\boldsymbol{x}^T \boldsymbol{g}) \tag{4}$$

These conjunctive $\boldsymbol{p}$ representations are stored in 'fast weights' [1]via Hebbian learning;

$$\boldsymbol{M}_t = \sum_{\tau=1}^{t-1} \boldsymbol{p}_\tau^T \boldsymbol{p}_\tau \tag{5}$$

And they can later be retrieved using an attractor network (a continuous version of the Hopfield network). Here a query vector $\boldsymbol{q}$ (details next paragraph) is inputted into the network and updated via;

$$\boldsymbol{q} \leftarrow \sigma(\boldsymbol{q}\boldsymbol{M}_t) \tag{6}$$

where $\sigma(\cdots)$ is a non-linear activation function; a ReLu in TEM. Crucially, because the memories are formed using both $\boldsymbol{g}$ and $\boldsymbol{x}$, they can be retrieved (pattern-completed) using just one of those representations alone i.e. 'what did I see the last time I was here' or 'where was I the last time I saw this'. To retrieve a memorised conjunction $\boldsymbol{p}$, TEM imagines (path-integrates) the next location $\boldsymbol{g}$ and provides this as input to the attractor network in the form $\boldsymbol{q} = flatten(\mathbb{1}^T \boldsymbol{g})$. Equation 6 is then iterated until a memory is retrieved.

Finally, to make sensory predictions, the retrieved conjunctive memory ($\boldsymbol{p}_t^{retrieved}$) is 'deconjunctified' into sensory and location components. The sensory component is obtained by unflattening $\boldsymbol{p}_t^{retrieved}$ and summing over the $\boldsymbol{g}$ dimension (Figure 8);

$$\boldsymbol{x}_t^{retrieved} = sum(unflatten(\boldsymbol{p}_t^{retrieved}), 1) \tag{7}$$

Finally, to make the sensory prediction $\boldsymbol{x}_t^{retrieved}$ is fed through a MLP $\boldsymbol{z}_t = f_\theta(\boldsymbol{x}_t^{retrieved}))$ to classify (predict) the upcoming sensory observation.

It is also possible, and often helpful, to project $\boldsymbol{g}$ and $\boldsymbol{x}$ via $\boldsymbol{W}_g$ and $\boldsymbol{W}_x$; $\tilde{\boldsymbol{g}} = \boldsymbol{g}\boldsymbol{W}_g$ and $\tilde{\boldsymbol{x}} = \boldsymbol{x}\boldsymbol{W}_x$ before they are combined conjunctively [2].

5 TEM AS A TRANSFORMER

Here we show that the above equations of TEM can be written so that: 1) the memory retrieval components looks like a transformer self-attention; 2) the path integration representation, $\boldsymbol{g}$ look like position encodings.

1) When considering the TEM memory retrieval process more closely (in this analysis, for direct comparison, we are only considering 1 attractor step in TEM with no non-linearity), we see that the attractor update $\boldsymbol{q}_t\boldsymbol{M}_t = \boldsymbol{q}_t \sum_\tau^t \boldsymbol{p}_\tau^T \boldsymbol{p}_\tau$ is simply equal to

$$\boldsymbol{p}_t^{retrieved} = \sum_\tau^t [\boldsymbol{q}_t \boldsymbol{p}_\tau^T] \boldsymbol{p}_\tau \tag{8}$$

[1]We note such 'fast weights' have previously been thought of as an alternative to the LSTM (Ba et al., 2016a).

[2]In fact, in the TEM code online, $\boldsymbol{W}_g$ and $\boldsymbol{W}_x$ are set as fixed weight matrices, where $\boldsymbol{W}_g$ sub-samples $\boldsymbol{g}$ and $\boldsymbol{W}_x$ transforms (compresses) $\boldsymbol{x}$ from a one-hot to a two-hot representation.

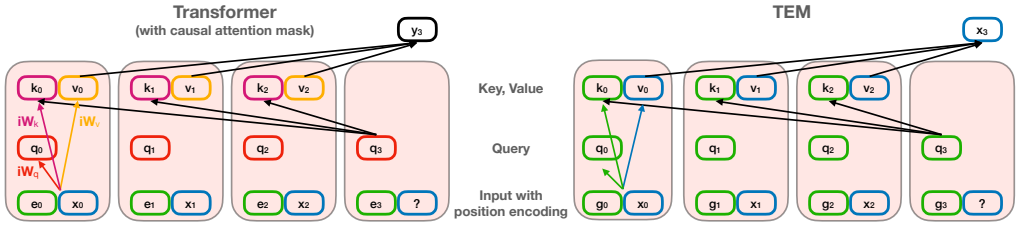

Figure 3: Self-attention in **(a)** Transformers and **(b)** TEM.

Since $[\boldsymbol{q}_t \boldsymbol{p}_\tau^T]$ is just a dot-product ($[\boldsymbol{q}_t \cdot \boldsymbol{p}_\tau]$), a single step of the attractor just retrieves memories weighted by their similarity (dot product) to the query. As noted by Ramsauer et al. (2020), this is exactly like a transformer but without the softmax scaling the dot-products. Thus the TEM memory retrieval process behaves like transformer self-attention.

2) We can however go further since TEM's input to the transformer (i.e. the TEM memories) are special; they are learnable and built from an outer product between $\tilde{\boldsymbol{g}}$ and $\tilde{\boldsymbol{x}}$ ($\boldsymbol{p}_\tau = flatten(\tilde{\boldsymbol{x}}_\tau^T \tilde{\boldsymbol{g}}_\tau)$), and these memories can be retrieved by a query based on $\tilde{\boldsymbol{g}}$ or $\tilde{\boldsymbol{x}}$ alone (e.g. $\boldsymbol{q}_t = flatten(\mathbb{1}^T \tilde{\boldsymbol{g}}_t)$). Together, these properties mean we can reduce the above dot product even further;

$$[\boldsymbol{q}_t \boldsymbol{p}_\tau^T] = \bar{\tilde{x}}_\tau [\tilde{\boldsymbol{g}}_t \cdot \tilde{\boldsymbol{g}}_\tau] \qquad \rightarrow \qquad \boldsymbol{p}_t^{retrieved} = \tilde{\boldsymbol{g}}_t \tilde{\boldsymbol{G}}^T \boldsymbol{\Lambda}_x \boldsymbol{P} \tag{9}$$

Where $\bar{\tilde{x}} = \sum_i (\tilde{\boldsymbol{x}}_\tau)_i$ and $\boldsymbol{\Lambda}_x$ is a diagonal matrix with elements $\bar{\tilde{x}}_\tau$ (see Appendix for an alternative derivation using vector elements). Thus to retrieve a conjunctive $\boldsymbol{p}$ memory, all that was necessary is weighting past $\boldsymbol{p}$ representations via 'self-attention' of $\tilde{\boldsymbol{g}}_t$ to past representations $\tilde{\boldsymbol{G}}$.

To simplify this even further, we consider what happens when we 'deconjunctify' $\boldsymbol{p}_t^{retrieved}$ to obtain the **sensory component of the memory**. Following the TEM procedure described above (Figure 8);

$$\tilde{\boldsymbol{x}}_t^{retrieved} = sum(unflatten(\boldsymbol{p}_t^{retrieved}), 1) = \sum_\tau^t \tilde{\boldsymbol{x}}_\tau \bar{\tilde{g}}_\tau \bar{\tilde{x}}_\tau [\tilde{\boldsymbol{g}}_t \cdot \tilde{\boldsymbol{g}}_\tau] = \tilde{\boldsymbol{g}}_t \tilde{\boldsymbol{G}}^T \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_x \tilde{\boldsymbol{X}} \tag{10}$$

Where $\boldsymbol{\Lambda}_g$ is a diagonal matrix with elements $\bar{\tilde{g}}_\tau = \sum_i (\tilde{\boldsymbol{g}}_\tau)_i$. Now all that is necessary to retrieve the sensory component of the memory is weighting past $\tilde{\boldsymbol{x}}$ representations with via 'self-attention' of $\tilde{\boldsymbol{g}}_t$ to past representations $\tilde{\boldsymbol{G}}$. This equation is now very similar to equation 1 except without the softmax and with additional weightings $\boldsymbol{\Lambda}_x$ and $\boldsymbol{\Lambda}_g$. These weighting however are likely learned to be constant (α) because otherwise some memories will be preferentially retrieved. In this case TEM is retrieving memories using

$$\tilde{\boldsymbol{x}}_t^{retrieved} = (\alpha \tilde{\boldsymbol{g}}_t \tilde{\boldsymbol{G}}^T) \tilde{\boldsymbol{X}} \qquad cf. \qquad softmax(\frac{\tilde{\boldsymbol{g}}_t \tilde{\boldsymbol{G}}^T}{\sqrt{d_k}}) \tilde{\boldsymbol{X}} \tag{11}$$

Which can be seen to be very closely related to the transformer equation (shown on the right), and diagrammatically shown in Figure 3. The model presented in this paper utilises the full transformer softmax rule.

The TEM-transformer. Thus the TEM-transformer (TEM-t; from Section 3) is this transformer that is directly analogous to TEM. Additional modelling details (analogous to modelling details in TEM) can be found in the Appendix. TEM-t offers dramatic performance improvements over the original TEM model (Figure 4; code will be released on publication). In particular, 1) Sample efficiency is increased; TEM-t requires many fewer data samples than TEM, and thus training time is reduced 2) TEM-t can tackle much larger problems, with the ability to store and retrieve many more memories (not shown here). Additionally to improved performance, TEM-t learns grid cells (Figure 1) and has potential implications for what place cells are (see next section).

Path integrating position encodings. This leads us to an interesting observation; we see that TEM's representations for path integration $\boldsymbol{g}$ plays the role of position encodings in transformers. However the structure of these positional encodings are not hard-coded, but instead *learned* via path integration (the structure of space!), with the particular position encoding depending on the particular

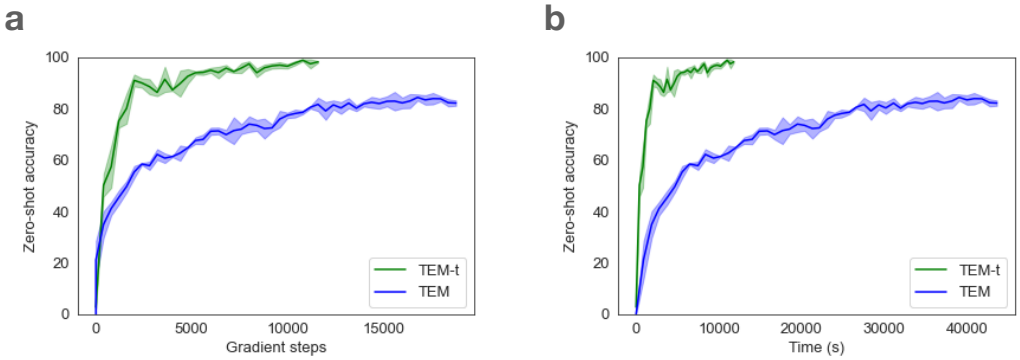

Figure 4: TEM-t is a more efficient learner than TEM, both in **(a)** sample efficiency and **(b)** time per gradient step. Zero-shot accuracy is prediction accuracy when taking links it has never taken before, but to a state it has visited before. Successful accuracy here is only possible with learned and generalised spatial knowledge. We have used the code from TEM from the TEM authors original code `https://github.com/djcrw/generalising-structural-knowledge`, and so have not optimised it for speed of learning etc, so we cannot claim this to be a fair comparison, nevertheless the difference is stark. We note that in the TEM paper, the authors say it takes up to 50,000 gradient updates for full training, whereas we stopped at 20,000.

sequence of actions taken. Other (non-spatial) structural representations could also be **learned** depending on the task at hand, i.e. grammar for language. This is a very different (and we think fruitful) re-understanding of position encodings; representing 'location' in a (learned) structure that can be inferred on the fly.

6 Place cells in transformers

Here we discuss, and demonstrate, how TEM-t offers a new interpretation of place representations. To do so we utilise a recent suggestion of how the transformer update can be performed in biological hardware (Krotov & Hopfield, 2020). In particular, self-attention (equation 1) can be split into two steps which correspond to two pools of neurons (Figure 5A); 1) calculate $softmax(\frac{\boldsymbol{q}_t \boldsymbol{K}^T}{\sqrt{d_k}})$. 2) multiply by $\boldsymbol{V}$. In this light, $\boldsymbol{K}$ and $\boldsymbol{V}$ can simply be seen as weight matrices between feature neuron (representing the query) and memory neurons (computing the softmax).

Since memory neurons are sparsely activated due to the softmax, they appear to have a spatial tuning for each environment resembling hippocampal place cells (Figure 5D-E; note Krotov & Hopfield (2020) stated memory neurons may correspond to place cells but without simulation). Similarly to experimentally recorded place cells, these neurons remap randomly between environments i.e. place cells being neighbours in one environment is not predictive of them being neighbours in another (unlike grid cells which maintain their phase neighbours across environments).

We can curate this architecture for the specifics of TEM-t. TEM-t explicitly considers factorised $\boldsymbol{g}$ and $\boldsymbol{x}$ representations (e.g. MEC and LEC), which project to feature neurons in hippocampus (or still in cortex). Thus the feature neurons consist of two separate sub-populations, $\tilde{\boldsymbol{g}} = \boldsymbol{g}\boldsymbol{W}_g$ and $\tilde{\boldsymbol{x}} = \boldsymbol{x}\boldsymbol{W}_x$, but which can connect to the same memory neurons in hippocampus (Figure 5B-C). These feature sub-populations are updated alternately rather than simultaneously, depending on the direction of retrieval; for example, when retrieving $\tilde{\boldsymbol{x}}$ the $\tilde{\boldsymbol{g}}$ feature neurons stay constant while the $\tilde{\boldsymbol{x}}$ neurons are updated (in turn updating $\boldsymbol{x}$ in LEC). In this vein, hippocampal memories link together cortical representations in potentially disparate brain areas. Thus TEM-t instantiates hippocampal indexing theory (Teyler & Rudy, 2007), which states that hippocampus provides an index that binds together cortical patterns across different brain regions.

The randomly remapping place cells described one paragraph ago cannot be the full picture since we know that place cell remapping is not random; instead individual place cells preferentially remap to locations consistent particular grid cell firing (as predicted by conjunctive memory cells $\boldsymbol{p}$ in TEM and verified experimentally in Whittington et al. (2020)). However another mechanism for this

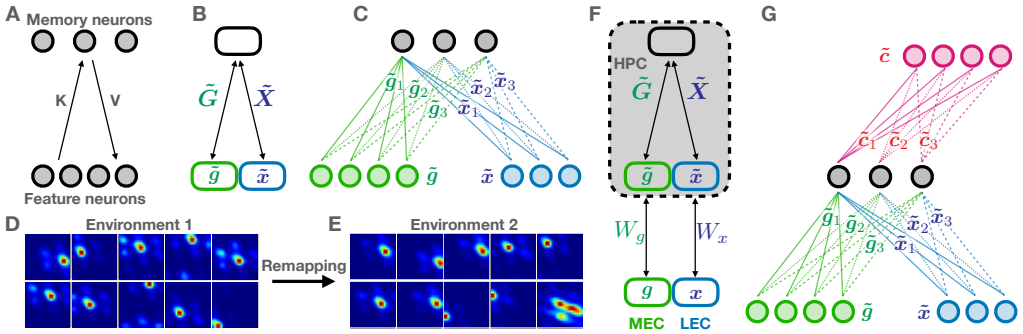

Figure 5: TEM-Transformer neural architecture. **(a)** Krotov & Hopfield (2020) describe a neurally plausible architectural instantiation the 'Hopfield networks is all you need' with a separation between 'feature' neurons (i.e. $\boldsymbol{h}$) and memory neurons (i.e. softmax($\boldsymbol{q}_t\boldsymbol{K}^T$). **(b-c)** This can be extended for TEM-t, but now the feature neurons are not all updated simultaneously, but only those across brain regions. **(d)** Memory neurons resemble hippocampal place cells and **(e)** remap randomly across environments. **(f)** A possible architecture where cortical neurons project to feature neurons in hippocampus which in turn project to memory neurons in hippocampus. **(g)** Additional brain regions can be included easily in this architecture with minimal increase in hippocampal neuron number.

phenomena born from TEM-t could be as follows. Should the feature neurons exist in hippocampus (Figure 5F) then there will be hippocampal spatial cells $\tilde{\boldsymbol{g}}$ that maintain their relationship to grid cells across different environment (as they are inherited from $\boldsymbol{g}$ via a projection $\tilde{\boldsymbol{g}} = \boldsymbol{g}\boldsymbol{W}_g$). Thus across the population of hippocampal cells, there will be those that maintain their relationship to grid cells (e.g. $\tilde{\boldsymbol{g}}$ and those that don't (e.g. memory neurons and $\tilde{\boldsymbol{x}}$), but the population effect will exist, just like what is experimentally observed.

As a note, the particular relationship of our model to the model of Krotov & Hopfield (2020), is what they refer to as a 'type B' model. These are models with contrastive normalisation on the memory neurons (via a softmax in our case), as opposed to 'type B' models which have a power activation function on the memory neurons. TEM (left hand side of Equation 11) corresponds to a linear activation function on the memory neurons, and is directly analogous to the original Hopfield energy. Secondly, the notion that there are two types of feature neurons that can be bound together in the same memory, was explored in Krotov & Hopfield (2016) where pixel intensities were associated with labels of those images. In TEM-t, one of the feature vectors, $\boldsymbol{g}$, is learned via a RNN and structures itself according to the underlying task structure.

As an additional aside, we note that Krotov & Hopfield (2020) architectures does not solve the scaling problem of conventional Hopfield networks; the number of memories that original Hopfield networks could store scaled linearly with the dimensionality of the recurrent attractor network (Amit et al., 1985). While recent analytical work has shown with exponential power activation functions, the number of memories that can be stored to scale as $2^{\frac{N}{2}}$, where $\{\mathrm{N}\}$ is the dimensionality of the feature neurons (Demircigil et al., 2017). This is a considerably more favourable scaling. However, unfortunately the architecture from Krotov & Hopfield (2020) instead requires a growing number of memory neurons (one for each memory), so the number of memories is still linear with the number of neurons! We note that mathematically derived scaling law was for an exponential activation function, not with a softmax as we use here.

7 Discussion

We have shown that TEM, a current model of the hippocampal formation, is closely related to a transformer with recurrent position encodings. We now consider some wider implications for neuroscience.

Multiple cortical inputs to hippocampus. TEM considers hippocampal conjunctions between two cortical regions ($\boldsymbol{g}$ and $\boldsymbol{x}$). It is, however, possible to consider conjunctions of more than two brain

regions. Indeed hippocampal neurons often respond to more than two task variables (McKenzie et al., 2014). In TEM, the naive approach of a 'triple' (or higher) conjunction would increase the number of hippocampal neurons would increase by a factor of n_c; the number of neurons from brain region $\tilde{\boldsymbol{c}}$. TEM-t does not scale so badly. Instead it just requires an additional n_c feature neurons, and the number of memory neurons can stay the same since the each hippocampal memory neuron can simply index a memory across three (or more), rather than two, brain regions (Figure 5G).

With multiple inputs to hippocampus $[\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{g}}, \tilde{\boldsymbol{c}}, \dots]$, any subset of those brain areas can reinstate a memory in the other brain regions i.e. $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{g}}$ can reinstate a $\tilde{\boldsymbol{c}}$ memory or $\tilde{\boldsymbol{g}}$ alone could reinstate $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{c}}$ memories. As an analogy to the TEM triple conjunction, TEM-t proposes that $\tilde{\boldsymbol{c}}_t$ is updated via $\tilde{\boldsymbol{c}}_t \leftarrow softmax((\tilde{\boldsymbol{g}}_t\tilde{\boldsymbol{G}}^T) \odot (\tilde{\boldsymbol{x}}_t\tilde{\boldsymbol{X}}^T))\tilde{\boldsymbol{C}}$, where $\odot$ is an element wise product. We note an alternate, and perhaps more intuitive, option could also be $\tilde{\boldsymbol{c}}_t = softmax(\tilde{\boldsymbol{g}}_t\tilde{\boldsymbol{G}}^T + \tilde{\boldsymbol{x}}_t\tilde{\boldsymbol{X}}^T)\tilde{\boldsymbol{C}}$.

Beyond hippocampus: Cortex as a Transformer. We have considered transformers as a model of hippocampus and its connections. We know, however, that transformer representations predict language areas (Schrimpf et al., 2020), and that patients can talk and comprehend just fine with major hippocampal deficits (Elward & Vargha-Khadem, 2018). This indicates that the transformer, and TEM-like models, may also model other brain regions, such as language areas, that are seemingly independent from hippocampus (related ideas discussed in Hawkins et al. (2019); Lewis (2021) but specifically for grid cells in neocortex). This raises two questions. Firstly what is the analogue of spatial positional encodings for higher order tasks such as language, and secondly what takes the role of the memory neurons if not hippocampus. We offer some thoughts in the following two paragraphs.

In spatial tasks, TEM and TEM-t learn positional encodings that mirror the structure of space. The implication is that positional encoding should reflect the abstract underlying properties of the task at hand. In language for example, this structure is grammar. This contrasts to the typical positional encodings in Transformers - sines and cosines - which represent a linear structure. It is our contention that positional encodings that are inferred on the fly and consist of previously learned structures (like the spatial case we have considered) would offer an interesting and potentially fruitful research direction in problems of language, maths, and logic.

If the transformer were solely instantiated in cortex, then what about the memory neurons? It is possible that the memory neuron equivalent exists in cortex too, but for these tasks, since it is not necessary to store long term memories or bind knowledge across multiple brain areas hippocampus is not required; so short term cortical memory neurons suffice.

8 Conclusion

We have shown that transformers with recurrent positional encodings reproduce neural representations found in rodent entorhinal cortex and hippocampus. We then showed these transformers are close cousins to models of hippocampus that neuroscientists have developed over the last few years. We hope this work brings neuroscience and machine learning closer together, and offers understanding for both sides; for neuroscientists a road map to understanding cortical areas beyond the hippocampal formation; for machine learners a greater understanding of positional encodings in transformers.

References

Daniel J. Amit, Hanoch Gutfreund, and H. Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55(14):1530–1533, 1985. ISSN 00319007. doi: 10.1103/PhysRevLett.55.1530.

Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using Fast Weights to Attend to the Recent Past. *Advances in Neural Information Processing Systems 29*, 29:4331–4339, 10 2016a. ISSN 10495258. URL http://arxiv.org/abs/1610.06258http://papers.nips.cc/paper/6057-using-fast-weights-to-attend-to-the-recent-past.pdf.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv preprint*, 2016b. ISSN 1607.06450. doi: 10.1038/nature14236. URL http://arxiv.org/abs/1607.06450.

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, 5 2018. ISSN 0028-0836. doi: 10.1038/s41586-018-0102-6. URL http://dx.doi.org/10.1038/s41586-018-0102-6http://www.nature.com/articles/s41586-018-0102-6.

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 preprint*, 2020. ISSN 23318422.

Yoram Burak and Ila R. Fiete. Accurate path integration in continuous attractor network models of grid cells. *PLoS Computational Biology*, 5(2):e1000291, 2 2009. ISSN 1553734X. doi: 10.1371/journal.pcbi.1000291. URL https://dx.plos.org/10.1371/journal.pcbi.1000291.

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*, 0:1–19, 3 2018. URL http://arxiv.org/abs/1803.07770.

Mete Demircigil, Judith Heusel, Matthias Löwe, Sven Upgang, and Franck Vermet. On a Model of Associative Memory with Huge Storage Capacity. *Journal of Statistical Physics*, 168(2):288–299, 2017. ISSN 00224715. doi: 10.1007/s10955-017-1806-y.

Yedidyah Dordek, Daniel Soudry, Ron Meir, and Dori Derdikman. Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis. *eLife*, 5 (MARCH2016):1–36, 2016. ISSN 2050084X. doi: 10.7554/eLife.10094.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint*, pp. 1–21, 2020. URL http://arxiv.org/abs/2010.11929.

Rachael L. Elward and Faraneh Vargha-Khadem. Semantic memory in developmental amnesia. *Neuroscience Letters*, 680(April):23–30, 2018. ISSN 18727972. doi: 10.1016/j.neulet.2018.04.040. URL https://doi.org/10.1016/j.neulet.2018.04.040.

Dileep George, Rajeev V. Rikhye, Nishad Gothoskar, J. Swaroop Guntupalli, Antoine Dedieu, and Miguel Lázaro-Gredilla. Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps. *Nature Communications*, 12(1):2392, 12 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-22559-5. URL http://dx.doi.org/10.1038/s41467-021-22559-5https://linkinghub.elsevier.com/retrieve/pii/B0123693985003418http://www.nature.com/articles/s41467-021-22559-5.

Nicholas J. Gustafson and Nathaniel D. Daw. Grid Cells, Place Cells, and Geodesic Generalization for Spatial Reinforcement Learning. *PLoS Computational Biology*, 7(10):e1002235, 10 2011. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1002235. URL https://dx.plos.org/10.1371/journal.pcbi.1002235.

Torkel Hafting, Marianne Fyhn, Sturla Molden, May-britt Britt Moser, and Edvard I. Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005. ISSN 00280836. doi: 10.1038/nature03721.

Jeff Hawkins, Marcus Lewis, Mirko Klukas, Scott Purdy, and Subutai Ahmad. A Framework for Intelligence and Cortical Function Based on Grid Cells in the Neocortex. *Frontiers in Neural Circuits*, 12(January):1–15, 1 2019. ISSN 1662-5110. doi: 10.3389/fncir.2018.00121. URL https://www.numenta.com/assets/pdf/research-publications/papers/Companion-paper-to-A-Framework-for-Intelligence-and-Cortical-Function-Based-on-Gri pdfhttps://www.frontiersin.org/article/10.3389/fncir.2018.00121/full.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 12 2016. URL http://arxiv.org/abs/1512.03385.

J J Hopfield. Neural networks and physical systems with emergent collective computational abilities (associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices). *Biophysics*, 79(April):2554–2558, 1982.

Seyed Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation. *PLoS Computational Biology*, 10(11), 2014. ISSN 15537358. doi: 10.1371/journal.pcbi.1003915.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pp. 1–9, 2012. ISSN 10495258. doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

Dmitry Krotov and John Hopfield. Large Associative Memory Problem in Neurobiology and Machine Learning. *arXiv preprint*, pp. 1–12, 2020. URL http://arxiv.org/abs/2008.06996.

Dmitry Krotov and John J. Hopfield. Dense Associative Memory for Pattern Recognition. *Advances in Neural Information Processing Systems*, 0(Nips):1180–1188, 6 2016. ISSN 10495258. URL http://arxiv.org/abs/1606.01164.

Julia Krupic, Neil Burgess, and John O'Keefe. Neural Representations of Location Composed of Spatially Periodic Bands. *Science*, 337(6096):853–857, 8 2012. ISSN 0036-8075. doi: 10.1126/science.1222403. URL http://www.sciencemag.org/content/337/6096/853.full.pdfhttps://www.sciencemag.org/lookup/doi/10.1126/science.1222403.

Yann Lecun, Le'on Bottou, Yoshua Bengio, and Parick Haffner. Gradient-based learning applied to document recognition. *proc. OF THE IEEE*, 1998.

Marcus Lewis. Hippocampal Spatial Mapping As Fast Graph Learning. *arXiv preprint*, 0, 7 2021. URL http://arxiv.org/abs/2107.00567.

Xuanqing Liu, Hsiang Fu Yu, Inderjit S. Dhillon, and Cho Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. *37th International Conference on Machine Learning, ICML 2020*, PartF16814:6283–6291, 2020.

Sam McKenzie, Andrea J. Frank, Nathaniel R. Kinsky, Blake Porter, Pamela D Rivie, Pamela D. Rivière, Howard Eichenbaum, Pamela D Rivie, Pamela D. Rivière, Howard Eichenbaum, Pamela D Rivie, Pamela D. Rivière, and Howard Eichenbaum. Hippocampal representation of related and opposing memories develop within distinct, hierarchically organized neural schemas. *Neuron*, 83(1):202–215, 7 2014. ISSN 10974199. doi: 10.1016/j.neuron.2014.05.019. URL https://linkinghub.elsevier.com/retrieve/pii/S089662731400405X.

John O'Keefe and J. Dostrovsky. The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171–175, 11 1971. ISSN 00068993. doi: 10.1016/0006-8993(71)90358-1. URL http://linkinghub.elsevier.com/retrieve/pii/0006899371903581.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. *arXiv preprint*, 2020. ISSN 23318422.

Honi Sanders, Matthew Wilson, Mirko Klukas, Sugandha Sharma, and Ila Fiete. Efficient Inference in Structured Spaces. *Cell*, 183(5):1147–1148, 11 2020. ISSN 00928674. doi: 10.1016/j.cell.2020.11.008. URL https://doi.org/10.1016/j.cell.2020.11.008https://linkinghub.elsevier.com/retrieve/pii/S0092867420315191.

Martin Schrimpf, Idan Blank, Greta Tuckute, Carina Kauf, Eghbal Hosseini, Nancy Kanwisher, Joshua Tenenbaum, and Evelina Fedorenko. The neural architecture of language: Integrative reverse-engineering converges on a model for predictive processing. *bioRxiv preprint*, 2020. doi: 10.1101/2020.06.26.174482.

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 32*, 32(NeurIPS):10003–10013, 2019.

Kimberley L. Kimberly L. Kimberley L. Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653, 2017. ISSN 15461726. doi: 10.1038/nn.4650.

Timothy J Teyler and Jerry W Rudy. The hippocampal indexing theory and episodic memory: Updating the index. *Hippocampus*, 17(12):1158–1169, 12 2007. ISSN 10509631. doi: 10.1002/hipo.20350. URL https://onlinelibrary.wiley.com/doi/10.1002/hipo.20350http://doi.wiley.com/10.1002/hipo.20350.

Benigno Uria, Borja Ibarz, Andrea Banino, Vinicius Zambaldi, Dharshan Kumaran, Demis Hassabis, Caswell Barry, and Charles Blundell. The spatial memory pipeline: A model of egocentric to allocentric understanding in mammalian brains. *bioRxiv preprint*, pp. 1–52, 2020. ISSN 26928205. doi: 10.1101/2020.11.11.378141.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):5999–6009, 2017. ISSN 10495258.

Zhiwei Wang, Yao Ma, Zitao Liu, and Jiliang Tang. R-Transformer: Recurrent Neural Network Enhanced Transformer. *arXiv preprint*, 0:1–11, 2019. URL http://arxiv.org/abs/1907.05572.

James CR R. Whittington, Timothy H. Muller, Shirley Mark, Caswell Barry, Neil Burgess, Timothy E.J. EJ Behrens, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy E.J. EJ Behrens. The Tolman-Eichenbaum Machine: Unifying Space and Relational Memory through Generalization in the Hippocampal Formation. *Cell*, 183(5):1249–1263, 11 2020. ISSN 00928674. doi: 10.1016/j.cell.2020.10.024. URL https://doi.org/10.1016/j.cell.2020.10.024https://linkinghub.elsevier.com/retrieve/pii/S009286742031388X.

Daniel L K Yamins, Ha Hong, Charles F. Cadieu, Ethan A. Solomon, Darren Seibert, and James J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences of the United States of America*, 111 (23):8619–8624, 2014. ISSN 10916490. doi: 10.1073/pnas.1403112111.

A Appendix

A.1 The maths using elements

Here we derive the main results using vector and matrix elements. Since each element in $\boldsymbol{p}$ is multiplicative combination of every pair of elements in $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{g}}$, then

$$p_{ij} = \tilde{g}_i \tilde{x}_j \tag{12}$$

Where we label the elements of vector $\boldsymbol{p}$ with two indices even though it is a vector. Similarly, the memory matrix uses 4 indices;

$$M_{ijkl} = \sum_{\tau} \tilde{g}_i^{\tau} \tilde{x}_j^{\tau} \tilde{g}_k^{\tau} \tilde{x}_l^{\tau} \tag{13}$$

Retrieving a memory via path integration. Querying the network becomes $q_{ij} = \tilde{g}_i^{PI}$, which reduces to

$$\begin{aligned} \sum_{ijk} q_{ij} M_{ijkl} &= \sum_{\tau} \sum_{ijk} \tilde{g}_i^{PI} \tilde{g}_i^{\tau} \tilde{x}_j^{\tau} \tilde{g}_k^{\tau} \tilde{x}_l^{\tau} \\ &= \sum_{\tau} \tilde{x}_l^{\tau} (\sum_k \tilde{g}_k^{\tau})(\sum_i \tilde{g}_i^{PI} \tilde{g}_i^{\tau})(\sum_j \tilde{x}_j^{\tau}) \\ &\rightarrow \tilde{\boldsymbol{g}}_t^{PI} \tilde{\boldsymbol{X}}^T \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_x \tilde{\boldsymbol{G}} \end{aligned} \tag{14}$$

Retrieving a memory using a sensory observation alone. Similarly when the query is the sensory observation alone $q_{ij} = x_j$

$$\begin{aligned} \sum_{ijl} q_{ij} M_{ijkl} &= \sum_{ijl\tau} \tilde{x}_j \tilde{g}_i^{\tau} \tilde{x}_j^{\tau} \tilde{g}_k^{\tau} \tilde{x}_l^{\tau} \\ &= \sum_{\tau} \tilde{g}_k^{\tau} (\sum_l \tilde{x}_l^{\tau})(\sum_i \tilde{g}_i^{\tau})(\sum_j \tilde{x}_j \tilde{x}_j^{\tau}) \\ &\rightarrow \tilde{\boldsymbol{x}}_t \tilde{\boldsymbol{X}}^T \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_x \tilde{\boldsymbol{G}} \end{aligned} \tag{15}$$

Retrieving a memory using a sensory observation and path integration. When the query is $q_{ij} = g_i^{PI} x_j$, and we want to retrieve a position encoding memory

$$\begin{aligned} \sum_{ijl} q_{ij} M_{ijkl} &= \sum_{ijl\tau} \tilde{g}_i^{PI} \tilde{x}_j \tilde{g}_i^{\tau} \tilde{x}_j^{\tau} \tilde{g}_k^{\tau} \tilde{x}_l^{\tau} \\ &= \sum_{\tau} \tilde{g}_k^{\tau} (\sum_l \tilde{x}_l^{\tau})(\sum_i \tilde{g}_i^{PI} \tilde{g}_i^{\tau})(\sum_j \tilde{x}_j \tilde{x}_j^{\tau}) \\ &\rightarrow (\tilde{\boldsymbol{x}}_t \tilde{\boldsymbol{X}}^T \odot \tilde{\boldsymbol{g}}_t^{PI} \tilde{\boldsymbol{G}}^T) \boldsymbol{\Lambda}_g \tilde{\boldsymbol{G}} \end{aligned} \tag{16}$$

A.2 Details of implementation

We will release code on publication.

Scaling beta. Since the number of memories is not constant, we use an adaptive β parameter in the softmax. This is because normalisation term in the softmax sums the number of memories, and so more memories down-weights probabilities. We want a self-attention value to not be affected by the number of elements in the set. In particular, we weight the softmax by $log(n_{memories})$.

Losses. We use same set of losses as in TEM;

- a one-step sensory prediction cross entropy loss, i.e. using $\boldsymbol{g}_t^{PI}$ as input to the memory retrieval process
- a sensory prediction cross entropy loss using $\boldsymbol{g}_t$ as input to the memory retrieval process
- a squared error loss between $\boldsymbol{g}_t$ and $\boldsymbol{g}_t^{PI}$
- l2 weight regularisation
- l2 regularisation on $\boldsymbol{g}_t$

Normalisation. We find that using layernorm (Ba et al., 2016b) on the positional encodings (not in the RNN, but on the input to transformer) to be beneficial, since the memory retrieval process can then be standardised - no one memory is up-weighted relative to others. Using layernorm before

self-attention in transformers is common practice (Vaswani et al., 2017). For simplicity, we use fixed weights on the layer norm, i.e. is is just a z score of $\boldsymbol{g}$, Since we use a one-hot encodings of our sensory representations, they are already normalised.

Stabilising position representations. Recurrently generated positional encodings accumulate noise and drift. While bespoke path integration networks from neuroscience mitigate noise by enforcing their neural representations to stay close to a neural manifold (Burak & Fiete, 2009), this can not be guaranteed in learned recurrent networks. One method of stabilisation is via sensory landmarks - i.e. 'what positional encoding did I have the last time I saw this landmark'. In this vein TEM uses the following query to the memory network; $\boldsymbol{q}_t = flatten(\tilde{\boldsymbol{x}}_t^T \mathbb{1})$, and **memories of positional encodings** are retrieved as;

$$\tilde{\boldsymbol{g}}_t^{retrieved} = sum(unflatten(\boldsymbol{q}_t \boldsymbol{M}_t), 0) = \sum_{\tau}^{t} \tilde{\boldsymbol{g}}_\tau \bar{\tilde{g}}_\tau \bar{\tilde{x}}_\tau [\tilde{\boldsymbol{x}}_t \cdot \tilde{\boldsymbol{x}}_\tau] = \tilde{\boldsymbol{x}}_t \tilde{\boldsymbol{X}}^T \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_x \tilde{\boldsymbol{G}} \tag{17}$$

The final position encoding, $\boldsymbol{g}$ is computed on a basis of path integration ($\boldsymbol{g}_t^{PI}$; equation 3) and stored landmark information ($\boldsymbol{g}_t^{retrieved}$; equation 17), i.e. $\boldsymbol{g}_t = \boldsymbol{g}_t^{PI} + f_\theta(\boldsymbol{g}_t^{retrieved}, \boldsymbol{g}_t^{PI})(\boldsymbol{g}_t^{retrieved} - \boldsymbol{g}_t^{PI})$, where $\boldsymbol{g}^{retrieved} = f_\theta(\tilde{\boldsymbol{g}}^{retrieved})$ and $f_\theta(\cdots)$ are different MLPs.

We note that a better query to the memory network would be $\boldsymbol{q}_t = flatten(\tilde{\boldsymbol{x}}_t^T \boldsymbol{g}_t^{PI})$ since sensory observations may be aliased, including $\boldsymbol{g}_t^{PI}$ can help disambiguate such aliasing. In this case, the retrieved memory is $(\tilde{\boldsymbol{x}}_t \boldsymbol{X}^T \odot \boldsymbol{g}_t^{PI} \boldsymbol{G}) \boldsymbol{\Lambda}_x \boldsymbol{G}$. This is perhaps, different to what would be anticipated; using an element wise product rather than an addition. Translating this other TEM memory retrieval process into transformers we get $\boldsymbol{g}_t^{retrieved} = f_\theta(softmax(\tilde{\boldsymbol{x}}_t \boldsymbol{X}^T \odot \boldsymbol{g}_t^{PI} \boldsymbol{G}) \boldsymbol{G})$, this can be thought of as another attention head i.e. from input $[\boldsymbol{g}, \boldsymbol{x}]$, the key and query 'attends' to $\boldsymbol{x}$ while the value 'attends' to $\boldsymbol{g}$.

When to add memories. Memories should not be added at every step, otherwise the self-attention mechanism would be biased towards memories (locations) that have been visited more frequently. This is not a problem typically addressed in transformer, but here to mitigate this issue memories are only added when existing memories for that particular conjunction do not already exist i.e. if there is a memory already with the present combination of $\boldsymbol{g}$ and $\boldsymbol{x}$ (similarity determined by dot product) then no new memory is added [3].

Multiple iterations: Though in our simulations we only used a single iteration of the transformer block, it is possible to use multiple iterations - just like a Hopfield network (as in Ramsauer et al. (2020)). Here the retrieved sensory memory $\tilde{\boldsymbol{x}}_t^{retrieved}$ can be fed into the next iteration, along with positional encoding i.e. the path integrated $\boldsymbol{g}$. In this case (see Appendix for derivation), TEM suggests that $\tilde{\boldsymbol{x}}_t$ is iteratively updated via $\tilde{\boldsymbol{x}}_t \leftarrow softmax(\tilde{\boldsymbol{x}}_t \tilde{\boldsymbol{X}}^T \odot \tilde{\boldsymbol{g}}_t \tilde{\boldsymbol{G}}) \tilde{\boldsymbol{X}}$, but with the first iteration via $\tilde{\boldsymbol{x}}_t \leftarrow softmax(\tilde{\boldsymbol{g}}_t \tilde{\boldsymbol{G}}) \tilde{\boldsymbol{X}}$. The multiplicative term in the softmax is perhaps unexpected - it may be thought that it should be additive instead.

Place-like representations without a softmax: Here we chose a softmax activation function on the memory neurons to make the relationship between TEM and transformers exact. However was this choice necessary to observe place cells in the memory neurons? While we have not examined this experimentally, we suspect place-like representations would emerge for a variety of activations, since memories must still be sparsely activated for correct prediction. We leave this for future work to verify, but note again that power and exponential activations have been shown to have a much greater memory capacity (Krotov & Hopfield, 2016; Demircigil et al., 2017) than traditional Hopfield nets with linear activation.

A.3 Additional details of the spatial task

We formalise a task type that not only relates to known hippocampal function, but also tests the learning and generalising of abstract structural knowledge. We formalise this via relational understanding on graph structures (a graph is a set of nodes that relate to each other).

[3] The TEM model actually does something very much like this - memories are only added if they did not already exist - $\boldsymbol{M} = \sum_\tau (\boldsymbol{p}_\tau - \hat{\boldsymbol{p}}_\tau)^T (\boldsymbol{p}_\tau + \hat{\boldsymbol{p}}_\tau)$, where $\hat{\boldsymbol{p}}_\tau$ is the memory that was retrieved at τ, i.e. only the un-predicted components of the memory $\boldsymbol{p}_\tau$ are added.

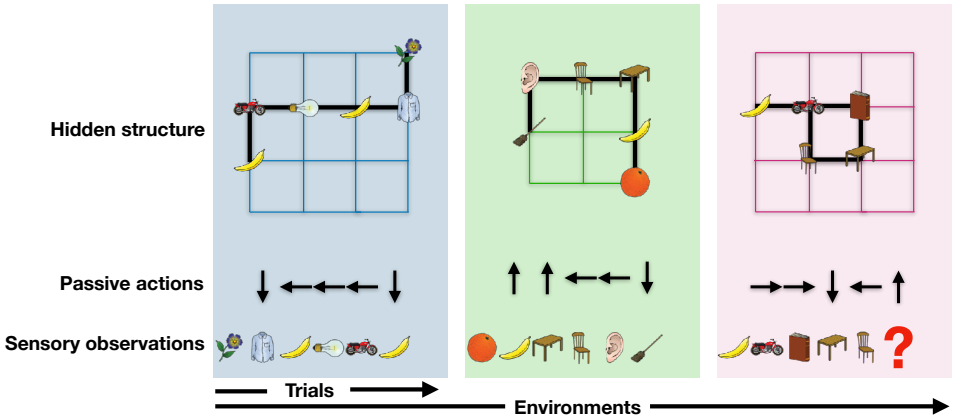

Figure 6: Learning to predict the next sensory observation in environments that share the same structure but differ in their sensory observations. TEM only sees the sensory observations and associated action taken, it is not told about the underlying structure - this must be learned.

Should one passively move on a graph (e.g. Figure 6), where each node is associated with a non-unique sensory observation (e.g. an image of a banana), then predicting the subsequent sensory observation tests whether you understand the graph structure you are in. For example, if you return to a previously visited node (Figure 6 pink) by a new direction - it is only possible to predict correctly if you know that a $right \rightarrow down \rightarrow left \rightarrow up$ means you're back in the same place. Knowledge of such loop closures is equivalent to understanding the structure of the graph.

We thus train our models on these graphs with it trying to predict the next sensory observation. Our models are trained on many environments sharing the same structure, e.g. 2D graphs (Figure 6), however the stimulus distribution is different (each vertex is randomly assigned a stimulus). Should it be able to learn and generalise this structural knowledge, then it should be able to enter new environments (structurally similar but with different stimulus distributions) and perform feats of loop closure on first presentation.

Formally, given data of the form $D = \{(\boldsymbol{x}^k_{\leq T}, \boldsymbol{a}^k_{\leq T})\}$ with $k \in \{1, \cdots, N\}$ (which environment it is in), where $\boldsymbol{x}_{\leq T}$ and $\boldsymbol{a}_{\leq T}$ are a sequence of sensory observations and associated actions/relations (Figure 6), N is the number of environments in the dataset, and T is the duration of time-steps in each environment, our model should maximise its probability of observing the sensory observations for each environment.

The sensory stimuli are chosen randomly, with replacement, at each node. We understand that this is not like the real world, where adjacent locations have sensory correlations - most notable in space (though names in a family tree will be less correlated). Sensory correlations help with sensory predictions, thus if we use environments with sensory correlations, we would not know what was causing the learned representations, sensory correlations, or transition structure. To answer this question cleanly, and to know that transition structure is the sole cause, we do not use environments with sensory correlations.

A.4 Additional details of TEM

We present a more detailed model schematic of TEM in Figure 7. We see there are two components to TEM - a RNN for understanding position ($\boldsymbol{g}$, in green top of Figure 7) that also indexes memories via 'queries' $\boldsymbol{q} = \boldsymbol{W}_g \boldsymbol{g}$. A memory network that binds together $\boldsymbol{x}$ and $\boldsymbol{g}$, via an outer product (middle green in 7, with more detail in Figure 8A). When the memory network is queried (red in Figure 7), it undergoes attractor dynamics to retrieve the full memory. To make a sensory prediction, the retrieved memory is 'deconjunctified' into a sensory representation (Figure 8C).

Model flow. TEM transitions through time and infers $\boldsymbol{g}_t$ and $\boldsymbol{p}_t$ at each time-step. $\boldsymbol{g}_t$ is inferred before forming each new memory $\boldsymbol{p}_t$. In other words variables $\boldsymbol{g}$ and $\boldsymbol{p}$ are inferred in the following order $\boldsymbol{g}_t, \boldsymbol{p}_t, \boldsymbol{g}_{t+1}, \boldsymbol{p}_{t+1}, \boldsymbol{g}_{t+2} \cdots$. This flow of information is shown in a schematic in Figure 7.

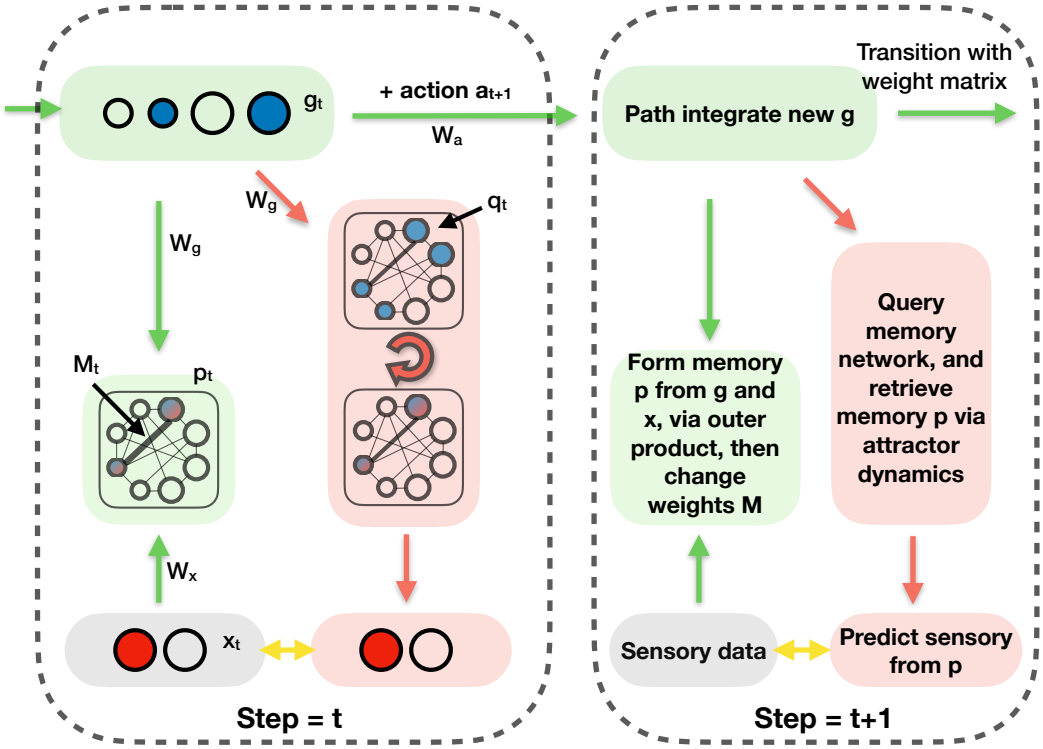

Figure 7: Schematic to show the model flow. Depiction of TEM at two time-points, with each time-point described at a different level of detail. Timepoint t shows network implementation, $t+1$ describes each computation in words. Red is for model predictions, green is for updating model variables. We do not show the stabilising position encodings module here. Circles depict neurons (blue is $\boldsymbol{g}$, red is $\boldsymbol{x}$, blue/red is $\boldsymbol{p}$); shaded boxes depict computation steps; arrows show learnable weights; looped arrows describe recurrent attractor. Black lines between neurons in attractor describe Hebbian weights $\boldsymbol{M}$. $\boldsymbol{W}_a$ are learnable, action dependent, transition weights. $\boldsymbol{W}_g$ and $\boldsymbol{W}_x$ are learnable projection matrices. Yellow arrows show training errors.

Independently, at each time-step, the model model asks 'are the inferred variables what I would have predicted given my current understanding of the world (weights)'. I.e. 1) Is the inferred $\boldsymbol{g}_t$ the one I would have predicted from $\boldsymbol{g}_{t-1}$. 2) Is the inferred $\boldsymbol{p}_t$ the one I would have predicted from $\boldsymbol{g}_t$. 3) Is $\boldsymbol{x}_t$ what I would have predicted from $\boldsymbol{p}_t$. This leads to errors (at each timestep) between inferred and predicted variables $\boldsymbol{g}_t$ and $\boldsymbol{p}_t$, and between sensory data $\boldsymbol{x}_t$ and its prediction.

At then end of a sequence, these errors are accumulated, with model parameters updated along the gradient (from back-propagation through time) that matches each others variables and also matches the data.

Since the model runs along uninterrupted, it's activity at one time-step influence those at later time-steps. Thus when learning (using back-propagation through time - BPTT), gradient information flows backwards in time. This is important as, should a bad memory be formed at one-time step, it will have consequences for later predictions - thus BPTT allows us to learn how to form memories and latent representations such that they will be useful many steps into the future.

A.5 Related work

Here we discuss and compare other models that recapitulate spatial representations found in the brain. These models fall into several categories. 1) Auto-encoder like models trained on spatial representations (Dordek et al., 2016). 2) Graph representation models (Gustafson & Daw, 2011; Stachenfeld et al., 2017). 3) Latent state inference models (George et al., 2021). 4) Path integrating

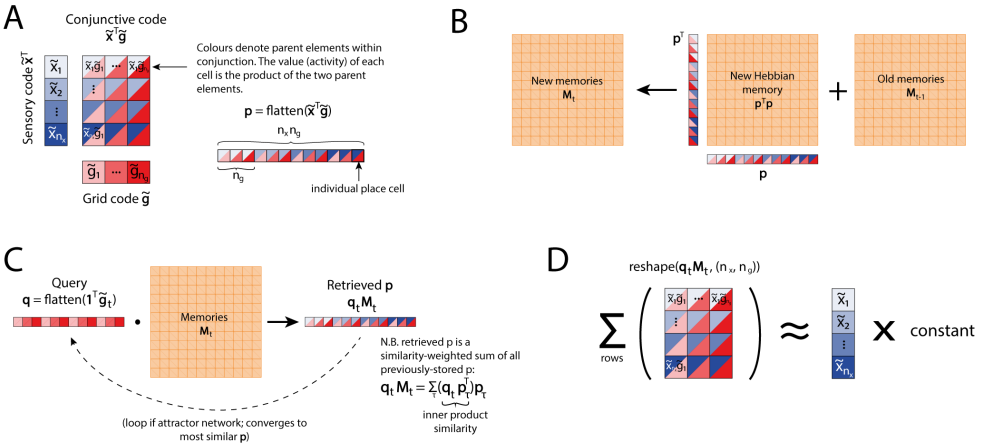

Figure 8: Memory formation and retrieval in TEM. **(a-b)** Memory formation. **(a)** Projected sensory sensory code $\tilde{\boldsymbol{x}}$ and projected grid code $\tilde{\boldsymbol{g}}$ are combined via an outer-product $\tilde{\boldsymbol{x}}^T\tilde{\boldsymbol{g}}$, which is flattened to obtain a vector of place cells $\boldsymbol{p}$. Each place cell (denoted by a single diagonally divided cell) is a conjunction of an element from each of $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{g}}$ (denoted by the two colours composing each cell). The activity of the place cell is the product of the values of these elements. **(b)** A new Hebbian memory $\boldsymbol{p}^T\boldsymbol{p}$ is added to the existing memory matrix $\boldsymbol{M}$. **(c-d)** Memory retrieval. **(c)** Multiplication of the query $\boldsymbol{q}$ with the memory matrix $\boldsymbol{M}$ retrieves a place code $\boldsymbol{p}$. This retrieved code is the sum of previously experienced codes, weighted by their similarity to the present query. This may be repeated iteratively to converge to the stored $\boldsymbol{p}$ that is most similar to $\boldsymbol{q}$. **(d)** The retrieved place code $\boldsymbol{p}$ is reshaped and summed along the rows to average-out the $\boldsymbol{g}$ components. The result is $\tilde{\bar{\boldsymbol{g}}}\tilde{\boldsymbol{x}}$.

models with RNNs trained on spatial representations (Cueva & Wei, 2018; Banino et al., 2018). 5) Models mixing RNNs and memory networks that are trained on sequences of sensory observations.

We note that models (1-4) are learned using curated spatial representation, i.e. the modeller has already worked how how space works, and the model is finding an alternate representation of that space. This is not how you or I learn. Instead, we learn by extracting regularities from the sensory world, and transfer/generalise this knowledge from domain to domain. Model category (5) does unsupervised learning by predicting sequences of sensory observations alone. From just sensory observations, it can slowly build up a picture of how space is structured, and then transfer this knowledge to situations that share the same underlying spatial structure. Our model TEM-t fall in the model (5) category, along with TEM (Whittington et al., 2020) and other similar models Uria et al. (2020).

For model to make sensory predictions as fast as possible, it is necessary to have two components. a) A RNN that understand position, and b) a memory network that can remember what you see and where you see them. Now the model can be asked what 'what will you see when heading `East`' and correctly respond `Cat` even it it has never gone East at that location before. This is because it can simulate going East via its RNN and then retrieve the memory it had stored at that location (it must have visited the Cat location before thought!). The other models, e.g. an autoencoder, could not solve this task since it learns slow statistical structures between sensory observations over many many training examples. Thus it would be clueless to the entirely new configuration of sensory observations in the new environment. In sum, **it is necessary to have an understanding of position, and the ability to make and retrieve memories to successfully make sensory predictions as fast as possible**.

We describe the relationship between TEM (and thus the hippocampal-entorhinal system) as close **not** because it produces the same representations, instead we are saying the relationship is close because **we have shown a mathematical relationship between the two models**. This could not happen for most ML models as most ML systems do not have both RNN structured representations and a memory system and so is not mathematically relatable to current models of the hippocampal

formation. However, equipped with the mathematical relationship, we can say that the memory part of the transformer is related to the hippocampal memory system in TEM. And the positional encodings are related to the entorhinal grid RNN system in TEM.

A.6 FURTHER LEARNED CELL REPRESENTATIONS

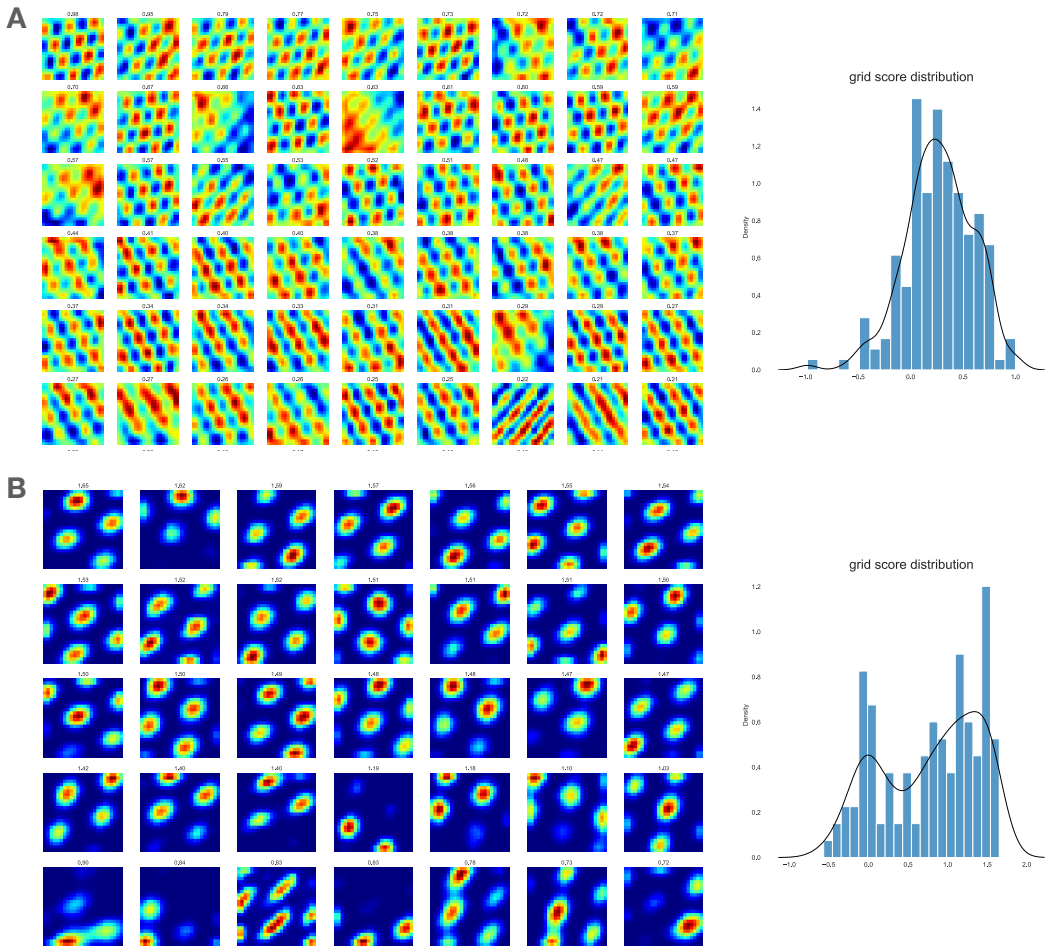

Figure 9: Learned grid cells ordered by grid score. We only show cells that are both active and have a grid score above 0. A gird score of 0.3-0.5 is generally considered to be a grid cell. The panels on the right hand side are the show the grid scores for the **whole population** of cells (though in some cases the grid score was not calculable e.g. if the cell has no activity; these cells are ignored in the analysis). **(a)** Grid cells learned with a linear activation function. **(a)** Grid cells learned with a ReLu activation function.

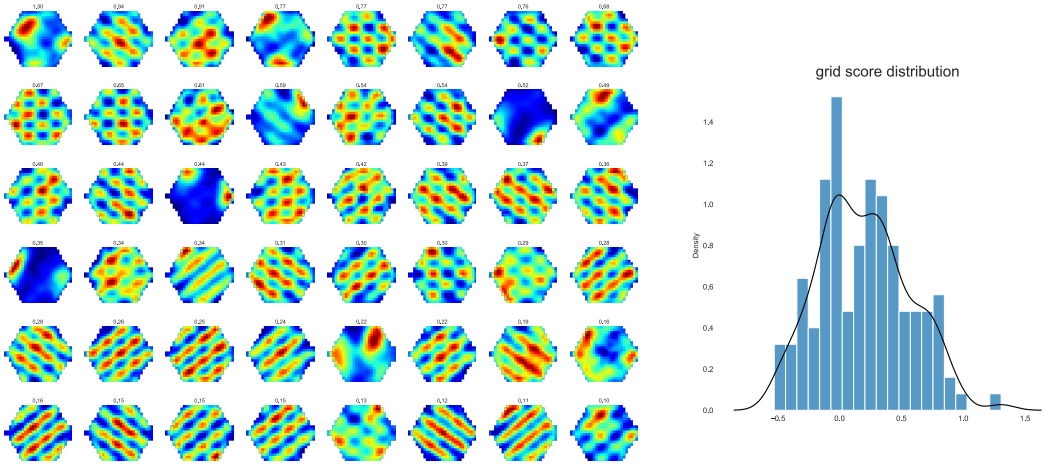

Figure 10: Learned grid cells in hexagonal 6-connected world.

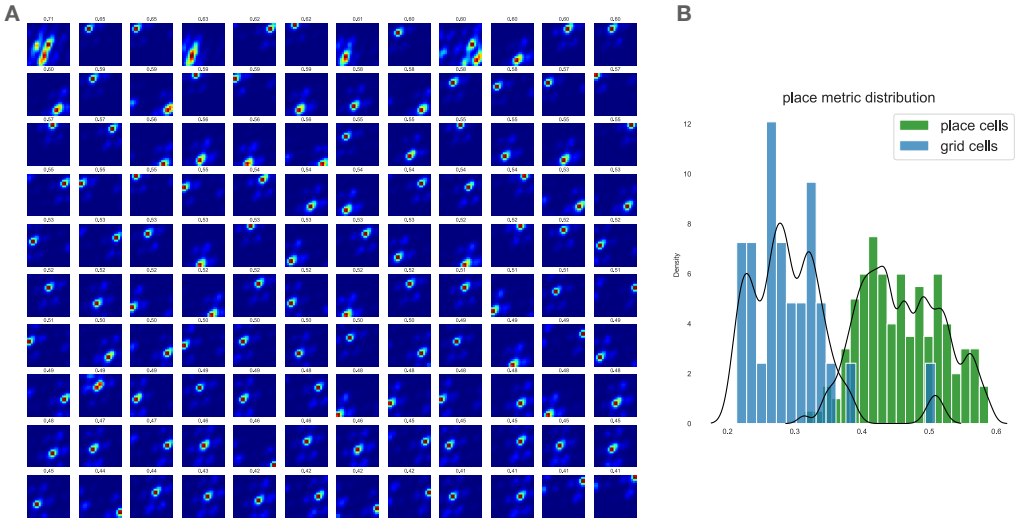

Figure 11: Place cells ordered by novel place cell metric. This metric assess how place-like the firing of each cell is. In particular, we look at the connected components of the firing rate map, and our metric is the ratio of 'firing mass' in the largest connected component versus all connected components. This metric is 1 if all the firing is in a single component, and it is lower if the firing is spread between components. **(a)** TEM-t learned memory place cells. **(b)** Our novel metric distinguishes between TEM-t RNN neurons (grid cells), and TEM-t memory neurons (place cells).