Rapid Learning without Catastrophic Forgetting in the Morris Water Maze

Raymond L Wang [1] **Jaedong Hwang** [1] **Akhilan Boopathy** [1] **Ila R Fiete** [1]

Abstract

Animals can swiftly adapt to novel tasks, while maintaining proficiency on previously trained tasks. This contrasts starkly with machine learning models, which struggle on these capabilities. We first propose a new task, the sequential Morris Water Maze (sWM), which extends a widely used task in the psychology and neuroscience fields and requires both rapid and continual learning. It has frequently been hypothesized that inductive biases from brains could help build better ML systems, but the addition of constraints typically hurts rather than helping ML performance. We draw inspiration from biology to show that combining 1) a content-addressable heteroassociative memory based on the entorhinal-hippocampal circuit with grid cells that retain shared across-environment structural representations and hippocampal cells that acquire environment-specific information; 2) a spatially invariant convolutional network architecture for rapid adaptation across unfamiliar environments; and 3) the ability to perform remapping, which orthogonalizes internal representations; leads to good generalization, rapid learning, and continual learning without forgetting, respectively. Our model outperforms ANN baselines from continual learning contexts applied to the task. It retains knowledge of past environments while rapidly acquiring the skills to navigate new ones, thereby addressing the seemingly opposing challenges of quick knowledge transfer and sustaining proficiency in previously learned tasks. These biologically motivated results may point the way toward ML algorithms with similar properties.

[1]Massachusetts Institute of Technology. Correspondence to: Raymond L Wang <rlwang@mit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Animals can *rapidly* learn new tasks that are conceptually similar to previously encountered tasks, but have different inputs and surface-level details; simultaneously, *they retain the ability to solve the previous tasks*. Neural modeling of this process of rapid conceptual knowledge transfer with retention of past learning has been limited. In some ways, rapid learning and learning retention seem to be in opposition: the former requires fast adaptation of parameters while the latter requires stable parameters. In machine learning, models tend to focus on either solving rapid learning and transfer, or on continual learning without forgetting; models tend not to do well at both.

Here, we build a biologically motivated neural model to solve a sequential version of the classic Morris Water Maze task (Morris, 1981; Vorhees & Williams, 2006), in which a rodent must find and then navigate to a submerged platform in a pool of cloudy water across multiple trials starting from different positions. We term our variant of this task the sequential Morris Water Maze (sWM) task. This task necessitates sequential learning across multiple unique environments, each characterized by a different platform location. Within a single environment, the task demands two generalizations from the agent. First, it must generalize its learning from a variety of starting locations. Second, it must rapidly adapt to changes in the goal locations. In our sequential version of the task, an additional layer of complexity is introduced. Here, the agent is required to learn new environments while preserving knowledge of the previous ones. This requirement tests the agent's ability to avoid catastrophic forgetting, a significant challenge in machine learning. Thus, the sWM task not only involves the aforementioned *intra-environment* generalization and adaptation but also *inter-environment* learning and memory retention.

Conventional unstructured neural networks suffer from catastrophic forgetting: a phenomenon where networks trained on a sequence of tasks fail to perform well on previously trained tasks (McCloskey & Cohen, 1989). Unstructured neural networks generally also lack an intrinsic ability to generalize rapidly to unseen tasks. Networks that perform rapid task transfer are typically extensively trained on a large number of related tasks (e.g. using multi-task learn-

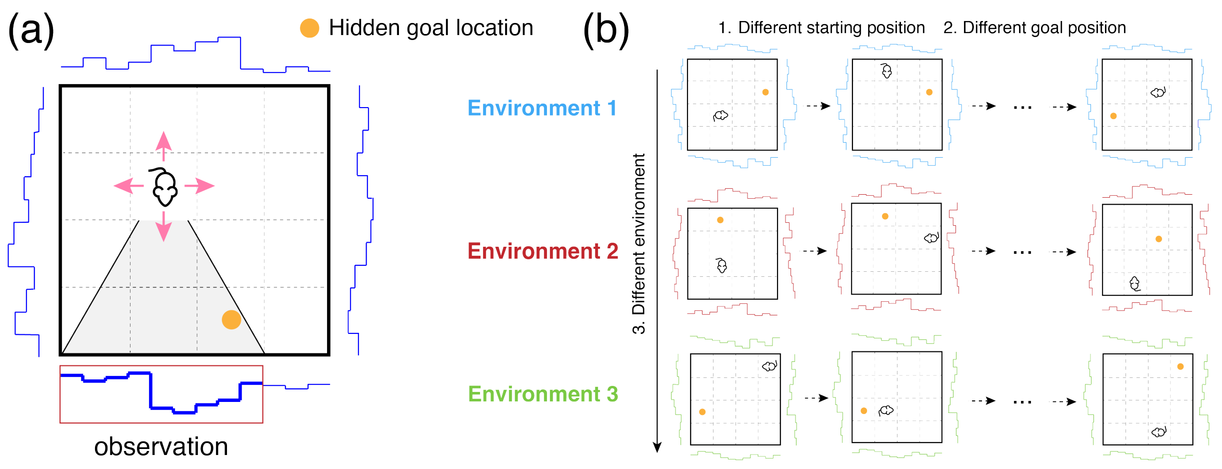

Figure 1. Schematic of the sequential Morris Water Maze task. **(a) The water maze environment.** The rodent icon represents the agent, arrows indicate the rodent's allowed actions, gold circles indicate the hidden platforms, and curves parallel to the walls of each environment denote patterns along the walls. The agent observes a portion of the wall pattern. **(b) our training setup.** Agents must generalize in three distinct ways. 1. Find a fixed goal location starting from random points in the environment. 2. Quickly learn new goal location within one environment, and reach it from random starting locations. 3. Learn various new environments, each with random start positions and multiple goal locations. The agent is evaluated on rapidly learning to navigate to new goal locations and in new environments and remembering navigation strategies from previously seen environments.

ing techniques (Caruana, 1997) or meta-learning (Thrun & Pratt, 1998)).

We hypothesize that certain inductive biases, like those present in the brain, allow networks to avoid these shortcomings and achieve performance comparable to animals on rapid and continual learning. It is known that animals exploit low-dimensional circuit dynamics in the entorhinal cortex and hippocampus to enable efficient spatial navigation and learning (O'Keefe & Dostrovsky, 1971; Hafting et al., 2005). We use a structured neocortical-entorhinal-hippocampal circuit based on a low-dimensional set of structured internal entorhinal states that do not vary across environments, the Vector Hippocampal Scaffolded Heteroassociative Memory (Vector-HaSH) architecture (Chandra et al., 2023), to enable such generalization in the Water Maze *after training only on a single environment*. Our model proceeds as follows: first, Vector-HaSH maps observation signals to a *grid cell* pattern, a type of spatial representation found in the entorhinal cortex. The grid code is then inputted into a randomly initialized, fixed Convolutional Neural Network (CNN), yielding a spatially invariant output feature vector. Lastly, this feature is processed by an attention module to determine the agent's action.

We integrate the high-capacity content-addressable memory system with a policy network that takes in the translationally invariant (grid) representations and computes a spatially invariant output vector that facilitates zero-shot policy learning in new environments. Conceptually, this combination is beneficial as it allows the system to store and retrieve relevant information efficiently, while also adapting rapidly to new environments without requiring additional training. This functionality reflects the learning behavior of biological entities, contributing to the agent's capacity for both knowledge retention and rapid, flexible learning. We emphasize that this is the first work to employ Vector-HaSH in continual learning tasks.

Our findings demonstrate a significant advantage of our neural-inspired method over general state-of-the-art continual learning algorithms in the sequential Morris Water Maze task. This outcome suggests that our method could offer valuable insights into developing more effective continual learning strategies, particularly in scenarios where traditional approaches fall short.

The contribution of this paper is threefold:

- We propose a new lifelong learning and rapid-learning task, the sequential Morris Water Maze (sWM), which extends the widely used Morris Water Maze test of spatial learning in animals.
- We propose a neuro-inspired lifelong learning algorithm based on Vector-HaSH (Chandra et al., 2023); the algorithm is designed to enable rapid learning while retaining knowledge over long time-scales.
- In sWM, our method achieves significantly higher per-

formance than baseline methods in standard continual learning.

2. Related Work

2.1. Continual Learning in Artificial Intelligence

Continual learning methods can be categorized into three approaches; 1) regularization-based methods, 2) replay-based methods, and 3) architecture-based methods. Regularization-based methods (Cheung et al., 2019; Kirkpatrick et al., 2017; Li & Hoiem, 2017; Zenke et al., 2017) employ regularization terms to constrain the changes in model parameters to preserve previous model weights. They balance the trade-off between stability and plasticity in the learning process. EWC (Kirkpatrick et al., 2017) leverages the Fisher information matrix to estimate an importance matrix used for parameter regularization so that the network can remember old parameters. LwF (Li & Hoiem, 2017) finds the output logits from an old model trained on a previous task and distills them into a new model. Replay-based methods (Robins, 1995) prevent forgetting by forming a replay buffer, a small exemplar set of previous data, or synthetic data (Van de Ven et al., 2020) to interleave with new tasks during training. Since the memory size is constrained, there are several approaches to find smaller subsets; reservoir sampling, reinforcement learning (Rebuffi et al., 2017), gradient-based selection (Aljundi et al., 2019). Another line of research employs existing sampling techniques and focuses on other aspects such as distillation (Douillard et al., 2020; Kang et al., 2022). Architecture-based methods focus on altering the model's *structure* to accommodate new tasks without affecting the performance of previous tasks. DEN (Yoon et al., 2018) dynamically expands neurons in the network. On the other hand, PNN (Rusu et al., 2016), DER (Yan et al., 2021) generates a new architectural backbone for each task, and FOSTER (Wang et al., 2022) distills a previous backbone network and a new backbone network into a single network applicable to the tasks corresponding to either backbone network.

2.2. Continual Learning in Neuroscience

Unlike continual learning with an artificial neural network, biological neural networks do not suffer from catastrophic forgetting (Morris, 1981). Aimone et al. (2010) argue that adult-born neurons contribute to learning new information while separating previous patterns. In the Morris Water Maze task, where a rodent navigates toward a hidden escape platform relying on distal cues, it directly heads to the platform even in an environment that was learned a few days ago (Morris, 1981; Vorhees & Williams, 2006). Place cells in the hippocampus play a key role in solving the task; they facilitate self-localization and route replay (Redish & Touretzky, 1998). Furthermore, they organize spatial information into separate maps when there is a significant shift in context or other non-spatial or spatial variables (*remapping*) (Colgin et al., 2008; Fyhn et al., 2007). This allows the rodent to remember each environment with associated platform location information, which enables it to navigate to the platform directly. Our method is based on Vector-HaSH (Chandra et al., 2023) which models the entorhinal-hippocampal circuit.

3. Morris Water Maze Task

We have developed a variant of the Morris Water Maze task called the sequential Morris Water Maze (sWM). This task assesses an artificial rodent's ability to remember previously explored environments while quickly learning new ones. In the original task, a rodent is placed in a circular tub filled with opaque fluid. Distal cues provide orienting and rough spatial information to the rodent. Inside the tank, there is a hidden platform that the rodent must find to avoid exhaustion from swimming. Once the rodent discovers the platform, the animal is placed in a different starting location within the same environment. This process is repeated multiple times. Then, the rodent is introduced to a different environment where the goal location and wall cues have changed, and the process repeats. Impressively, even after training in multiple subsequent environments, the rodent retains knowledge of previous environments and rapidly navigates toward the hidden platform.

For our task, we simplified the setup by using a square tub with distinctive markings on the walls as cues. These markings help the agent localize itself within the environment. The agent's objective is to efficiently locate a hidden platform within the environment. The agent can choose to take steps from among the four cardinal directions - north, south, east, or west.

Once the agent has been sufficiently trained in one environment, we introduce a sequential training regime. In this phase, the agent is exposed to both familiar and unfamiliar environments, with different starting points in each. Varying the starting points adds complexity, and requires the agent to adapt its strategies based on its current position and the goal location.

Our task provides a comprehensive evaluation of the agent's cognitive abilities, specifically focusing on its capacity to retain knowledge from past experiences and its ability to quickly learn from new ones. These are qualities that biological entities, like rodents, naturally possess and demonstrate with remarkable efficiency. By replicating these attributes in our artificial agent, we aim to create a system capable of navigating complex tasks with similar adeptness.

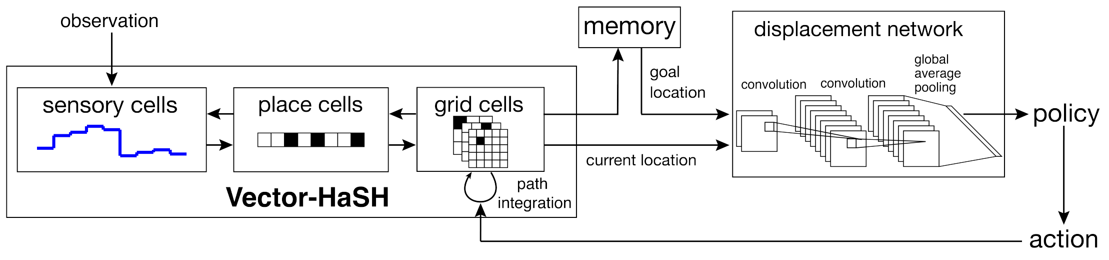

Figure 2. Schematic of our model: Vector-HaSH for Spatial Navigation (vHSN) The agent observes a portion of wall. Observations, along with velocity inputs, are fed into a Vector-HaSH network that produces grid cell activations representing the agent's location. An external memory module stores the grid code of the goal location. Grid codes of the current location and goal location are fed to the displacement network, a spatially-invariant convolutional neural network to produce a representation of the relative goal position. This is fed to a policy that produces actions.

4. Vector-HaSH

Vector-HaSH (Chandra et al., 2023) is a content-addressable memory (CAM) model based on the architecture of the neocortical-entorhinal-hippocampal memory circuit in the brain. Content-addressable memory models are networks that can store vectors (patterns to be memorized) as fixed points of their dynamics, and thereby recall/reconstruct them from noisy cues. Specifically, given a corrupted version of a previously encountered pattern, CAM models aim to reconstruct the original un-corrupted ground truth pattern. CAM models often suffer from a memory cliff problem: when the number of stored patterns crosses a certain threshold, the model not only fails to learn any new patterns, but also abruptly fails to recall all previously stored patterns. This is a form of *catastrophic forgetting*.

Vector-HaSH addresses the memory cliff problem by constructing a fixed scaffold of pre-defined content-independent fixed points, which are then used to store the content-laden patterns through hetero-associative learning, thus mimicking the neocortical-entorhinal-hippocampal circuit to store patterns. The Vector-HaSH architecture consists of three layers; features, hidden states, and labels, which biologically correspond to sensory input, place cell layer, and grid cell layer, respectively. We use grid code as labels instead of the k-hot labels in Vector-HaSH. The place cell layer $\mathbf{p} \in \{-1, +1\}^{N_P}$ represents an N_P dimensional binary vector, the grid cell layer $\mathbf{g} \in \{0, 1\}^{\sum \lambda_i}$ is defined as the concatenation of λ_i dimensional one-hot vectors each of which represents a grid module in the brain, and the sensory layer is N_s dimensional.

Before starting experiments, the memory scaffold (grid and place cells states, as well as the projections between the grid and place cell layers) is *pre-defined*. The projection matrix from the grid cell layer to the place cell layer, $\mathbf{W}_{PG}$ is randomly generated so that it maintains an injective projection. On the other hand, the weight matrix from the place cell layer to the grid cell layer is trained by Hebbian learning such that it associates each active place cell (defining a place code) to the concurrently active grid cells (defining a corresponding grid code):

$$\mathbf{W}_{GP} = \frac{1}{|\mathbf{N}|} \sum_{\mu=1}^{\mu=N} \mathbf{g} \cdot (\mathsf{sign}(\mathbf{W}_{PG} \cdot \mathbf{g}))^T, \tag{1}$$

where N is the number of training patterns.

When the agent explores the environment, the weights between sensory inputs and the place cells ($\mathbf{W}_{SP}$ and $\mathbf{W}_{PS}$) are learned by a pseudoinverse learning rule (Personnaz et al., 1985) in an online manner (Tapson & van Schaik, 2013), yielding the following final weights:

$$\mathbf{W}_{SP} = \mathbf{S} \cdot \mathbf{P}^\dagger, \tag{2}$$

$$\mathbf{W}_{PS} = \mathbf{P} \cdot \mathbf{S}^\dagger, \tag{3}$$

where $\mathbf{S}$ and $\mathbf{P}$ are $N_s \times N$ and $Np \times N$ dimensional matrices of sensory patterns and place patterns respectively, and $\dagger$ indicates the pseudoinverse.

In summary, given the sensory input $\mathbf{s}_t$ at time t, the corresponding place cell and grid cell activations are computed through the model dynamics as follows:

$$\mathbf{p}_t = \mathsf{sign}(\mathbf{W}_{PS} \cdot \mathbf{s}_t), \tag{4}$$

$$\mathbf{g}_t = \mathsf{CAN}(\mathbf{W}_{GP} \cdot \mathbf{p}_t). \tag{5}$$

where $\mathsf{CAN}(\cdot)$ represents the continuous attractor recurrence in the grid layer that is implemented using module-wise winner-take-all dynamics. This ensures that the equilibrium grid state is always a valid grid code i.e., a concatenation of one-hot vectors corresponding to each grid module.

The grid cell layer receives velocity signals (action input $\mathbf{a}_t$) for path integration, where the activated index for each grid cell module is shifted according to the action direction to infer the next grid state. Once we obtain the next grid code $\mathbf{g}_{t+1}$, its corresponding place code $\mathbf{p}_{t+1}$ is associated with the sensory input ($\mathbf{s}_{t+1}$).

In our implementation, we extend the grid cell modules to 2D space (with λ_i^2 dimensions for each one-hot grid cell module) and adapt the path integration described above to suit the proposed 2D sequential Morris Water Maze environments.

5. Vector-HaSH for Spatial Navigation (vHSN)

5.1. Motivation and Overview

Artificial neural networks, despite their significant advancements, are still prone to a major shortcoming known as 'catastrophic forgetting' during continual learning. This issue arises when these networks, after being trained on new tasks, tend to forget the old ones, thereby undermining their learning continuity. By contrast, natural organisms like rodents and humans showcase a remarkable resilience to such forgetting. This ability to continuously learn and adapt without forgetting past learning underscores the sophistication of biological learning systems. A wealth of scientific research has demonstrated that specific types of neurons, known as grid and place cells, play instrumental roles in counteracting catastrophic forgetting, particularly in the context of spatial memory. These cells, predominantly found in the hippocampus, are believed to create cognitive maps of the environment, helping the organism to navigate and remember spatial information.

Inspired by this, we design a novel method for continual learning based on Vector-HaSH (Chandra et al., 2023) called Vector-HaSH for Spatial Navigation (vHSN). To begin with, the Vector-HaSH converts observations into grid cell patterns (*grid code*). This involves representing the acquired data in a structured format that mimics the function of grid cells in the brain, which are integral to understanding spatial positioning and navigation. Next, the grid code is inputted into a randomly initialized fixed Convolutional Neural Network (CNN) to leverage its inherent spatial invariance, ensuring consistent output regardless of shifting inputs. Finally, an attention module takes the feature vector and retrieves the appropriate action based on features that have been observed previously.

5.2. Associating Grid Code Displacements with Movements

We develop a model of how rodents rapidly learn to navigate in new environments. Using a randomly initialized fixed convolutional neural network (CNN), our model maps the rodent's current and goal locations (encoded in a grid code, provided by Vector-HaSH) to a spatially-invariant representation of the *displacement* of the goal relative to its current position. We use an attention mechanism with the keys being the spatially invariant representation of the grid code and the values being the appropriate actions. During the training phase, these key-value pairs are associated and stored within the mechanism. During the testing phase, the agent's current state generates a spatially invariant representation of displacement that is used as the query. This query is then processed through a dot product operation with the existing keys in the dictionary. The action associated with the key most similar to our query is identified and used. This process allows for efficient action selection based on the spatially invariant displacement of the agent. Our architecture's spatial invariance allows the agent to rapidly learn to navigate in unseen environments *by only learning associations between new observations and the grid code* (it does not need to learn new associations to actions) as we will discuss in the next section. Figure 2 illustrates our model architecture. In our appendix, Algorithm 1 illustrates the pseudocode of our training loop, while Algorithm 2 illustrates how our agent is updated, and Algorithm 3 illustrates how we use Vector-HaSH.

5.3. Agent Training and Zero-Shot Policy Learning in Novel Environments

The agent under consideration is now equipped with two crucial functionalities: the ability to counteract catastrophic forgetting and the capacity to facilitate forward transfer to novel environments. These two attributes together expedite the learning process.

During the initial phase of training, the agent is introduced to a novel environment where it initiates exploration. Concurrently, it collects observational data, forming associations between these observations and a grid code via the Vector-HaSH framework. This process effectively constructs a memory scaffold, enabling the agent to effectively navigate within a specific environmental context.

Upon successful identification of the goal within the environment, the agent proceeds to store the corresponding grid code. This stored grid code, signifying the goal location, serves as a key reference point in the agent's cognitive map of the environment.

Subsequently, from multiple locations within the environment, we use our spatially invariant CNN to compute a representation of the vector displacement between the agent's location and the pre-stored goal location. This displacement vector encapsulates the navigational 'distance' the agent must traverse to reach the goal from its current position.

These displacement representations are then processed by the attention mechanism. The mechanism associates displacement with the corresponding movement action required to progress toward the goal. Storing these associations allows the agent to retrieve the appropriate action when a previously observed displacement is encountered later.

Upon introduction to a new environment, the agent embarks

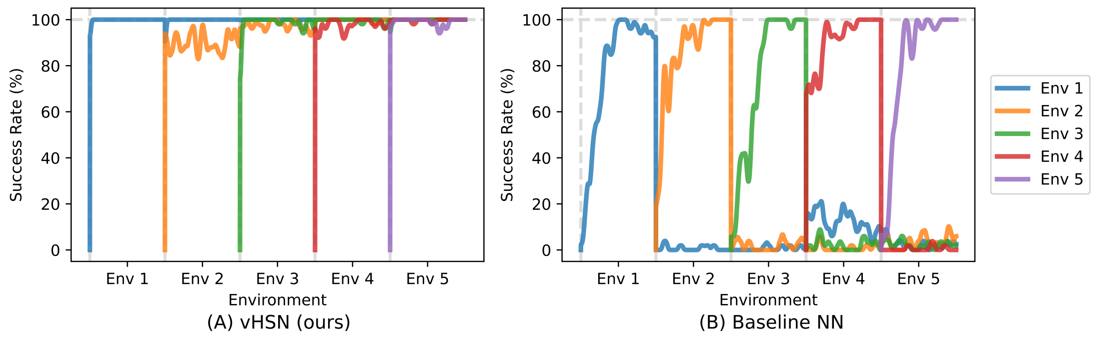

Figure 3. **Our model performance remains steady with the addition of new environments and avoids catastrophic forgetting.** The average success rate of each environment while training on the following environments of our final model (a) and ours without Vector-HaSH (b). The full model rapidly outperforms the one without Vector-HaSH. In both plots, we use a moving average of 25 points and Gaussian smoothing with $\sigma = 10$.

on a similar exploration phase. Once the goal is located in this new environment, a significant feature of our system emerges: *the policy requires no further training.* The agent first computes a representation of goal displacement using the spatially invariant CNN. Then, it applies the attention mechanism to the stored associations between displacement representations and movements to retrieve the correct navigational action.

This unique process facilitates zero-shot policy learning in new environments, underscoring the effectiveness and adaptability of our proposed framework. It exemplifies our agent's capacity to rapidly assimilate and apply knowledge, enabling successful navigation in unfamiliar environments.

6. Experiments

6.1. Experimental Details

We optimize parameters using Adam (Kingma & Ba, 2015) with a learning rate of 0.001 for 800 episodes for each environment. The maximum number of steps in each episode is set to 100 and the starting configuration (head direction and coordinates) are different. The environment is a 30×30 grid with unique, noise-added step function markings on the walls. The agent has a field of view (FOV) of 120 degrees (see Figure 1a). We use a public continual learning implementation (Zhou et al., 2023) for EWC (Kirkpatrick et al., 2017), another public continual learning implementation (Boschini et al., 2022) for DER (Buzzega et al., 2020), DER++ (Buzzega et al., 2020), A-GEM (Chaudhry et al., 2019), ER (Rolnick et al., 2019), and implement our own version of replay buffer and fine-tuning. For fine-tuning, we sequentially train on each environment. In our replay buffer implementation, we allocated a fixed buffer size (100 in our case) during the training of the neural network within a single environment. Throughout this training phase, we stochastically selected data points for inclusion in our replay buffer. Upon completion of training in one environment, we initiated a fine-tuning process on our replay buffer by sampling from it, followed by an evaluation in all previously trained environments. This procedure was replicated across all five environments. We selected the replay buffer size based on the performance. Details about (Boschini et al., 2022) implementation in our environment is included in the appendix.

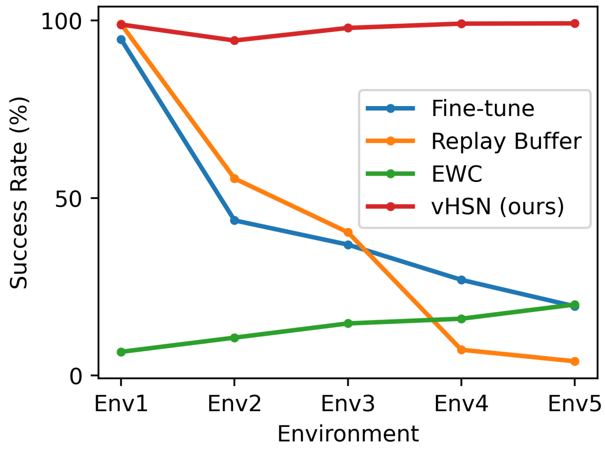

Figure 4. **Our method clearly outperforms the continual learning baselines, maintaining almost perfect performance, while other methods are degraded as training goes on.** We show the average success rate along incremental stages by testing on environments shown previously.

6.2. Comparison with Baselines

In Figure 3, our approach (a) exhibits rapid learning in the first environment compared to the baseline neural network trained in a fine-tuning framework shown in (b), where the observations are fed directly into a neural network and supervised by the correct action. Furthermore, our method successfully acquires a general, transferable navigation pol-

Table 1. The average success rate (%) of each environment after training on all environments. Our model maintains high success rates across all environments. In contrast, other methods perform poorly due to catastrophic forgetting, except for some methods in the last environment (Env 5).

Method	Average	Env 1	Env 2	Env 3	Env 4	Env 5
Fine-Tune	19.5	2.2	3.6	2.4	0.5	99.9
EWC (Kirkpatrick et al., 2017)	23.2	0.0	0.0	16.0	0.0	**100.0**
Replay Buffer	4.0	0.0	9.0	0.0	3.0	8.0
DER (Buzzega et al., 2020)	2.05	0.0	7.69	2.56	0.0	0.0
DER++ (Buzzega et al., 2020)	3.39	0.0	3.85	0.0	11.54	1.54
A-GEM (Chaudhry et al., 2019)	7.31	0.0	7.69	15.38	13.46	0.0
ER (Rolnick et al., 2019)	16.92	23.08	26.92	30.77	3.85	0.0
Ours(vHSN)	**99.2**	**99.2**	**99.5**	**99.0**	**99.5**	98.8

icy from this initial environment, allowing rapid navigation in subsequent environments *without any policy training*. This contributes to the prevention of catastrophic forgetting, as past environments can be recalled after recognizing the current environment through a few trajectories. In sharp contrast, the baseline experiments demonstrate an almost immediate onset of catastrophic forgetting upon exposure to a new environment. This phenomenon is marked by a rapid performance decline following the training of a few new trajectories, despite the initial successful knowledge transfer and adequate performance in the new setting.

To address this shortcoming of the baseline, we employed additional strategies in continual learning such as the use of a replay buffer and Elastic Weight Consolidation (EWC) on the baseline neural network. Despite these efforts, both the replay buffer strategy and EWC demonstrated signs of catastrophic forgetting. Figure 4 displays the average success rate of all previously trained environments after training on the environment indicated on the x-axis. Our method consistently outperforms the continual learning baselines, whereas other methods exhibit degraded performance as more environments are introduced for training. To benchmark against contemporary state-of-the-art methods, we selected four additional techniques: Dark Experience Replay (Buzzega et al., 2020), Dark Experience Replay ++ (Buzzega et al., 2020), A-GEM (Chaudhry et al., 2019), and Experience Replay (Rolnick et al., 2019). Table 1 shows our method compared to baselines on all five environments after all training is complete.

The underwhelming performance of EWC in our tasks appears to stem from the similarity of inputs across different environments. Despite these similarities, the goal positions differ between environments. Consequently, similar observations could map to two distinct actions. EWC aims to maintain the weights of the network to find an overlap between all tasks. However, due to this subtle complexity in our task design, EWC fails to perform optimally.

The underwhelming performance of the replay methods in our tasks appears to stem from the impact a few trajectories have on the network due to the similarity of the inputs in the environment. In addition, an observation in one environment that might be similar to an observation in another could map to different actions leading to interference.

6.3. Ablation Study

The effectiveness of each individual component in our proposed method is analyzed and summarized in Table 2 in the Appendix. Overall, the attention module plays a crucial role in achieving high performance. In fact, when used alone, the attention module achieves a perfect success rate. This is because when the goal location remains fixed, there is no need to rely on the spatial invariance provided by CNN (policy). Instead, the grid code can be directly associated with the attention module. This approach must learn associations between *every* observation and the corresponding ground-truth actions, which is memory inefficient and non-transferable to new environments. Furthermore, this approach becomes vulnerable when there are changes in the goal locations within the same environment since associations between observations and actions must be re-learned. On the other hand, when the attention module is combined with Vector-HaSH without CNN, the performance is significantly lower. This is likely because Vector-HaSH lacks spatial invariance, leading to the learning of conflicting associations between the grid code and actions. As for the CNN without the attention module, it corresponds to the ”Fine-Tuning” model presented in Table 1. The use of Vector-HaSH allows the CNN to use different input encoding methods, enhancing its versatility.

In summary, the superior accuracy demonstrated by the encoding network with attention, or by the attention mechanism in isolation, can primarily be attributed to its perfect memorization capabilities. This becomes apparent when the attention mechanism undergoes training as it is just storing key-value pairs. However, in the absence of such training, the model's performance in future environments

Table 2. **Combined components of vHSN allow for zero-shot transfer to new environments.** The last row is our final model. We measure the average success rate (%) across all environments after training the last environment. The exception is the case of vHSN: Vector-HaSH, the encoding network, and attention mechanism, where the system is trained in a single environment and subsequently tested across five different environments.

VECTOR-HASH	DISPLACEMENT NETWORK (DN)	ATTENTION	TRAINING ENVIRONMENT ID	SUCCESS RATE
		✓	1, 2, 3, 4, 5	100
	✓		1, 2, 3, 4, 5	19.5
	✓	✓	1, 2, 3, 4, 5	99.7
✓	✓		1, 2, 3, 4, 5	0.9
✓		✓	1, 2, 3, 4, 5	5.4
✓	✓	✓	1	99.2

Table 3. **Our model allows for adaptation to new goal locations not included during training.** The last row is our final model, vHSN. We measure the average success rate (%) of a new goal location after training the new goal in one environment. The exception is the case of Vector-HaSH, the encoding network, and attention mechanism, where the system is trained in a single environment and subsequently tested across five different environments.

VECTOR-HASH	DN	ATTENTION	SUCCESS RATE
		✓	1.6
	✓		2
	✓	✓	1.8
✓	✓		1
✓		✓	5.6
✓	✓(FC)	✓	0.1
✓	✓	✓	99.5

significantly declines, often nearing zero. This reveals a lack of forward transfer or generalizability in the model.

To verify the effectiveness of spatial invariance from CNN, we train one environment with the fixed goal location and evaluated it with the changed goal location. Table 3 shows that all three modules should be combined together to solve the new location with further training. Furthermore, we also test that using fully-connected layers (FC) instead of CNN cannot solve the problem, which emphasizes the need to spatial invariance to find unseen goal locations.

Our framework, which includes Vector-HaSH, the encoding network, and the attention mechanism, is trained exclusively on one environment and subsequently evaluated on four *unseen* environments and one seen environment. Conversely, all other ablated models are trained and then evaluated in all five environments. We adopted this strategy due to the observation that, without any training in future environments, each of our ablation study networks merely exhibited random movement, demonstrating no ability to generalize.

7. Discussion

We introduced a novel neural model, powered by the Vector-HaSH architecture, which exhibits remarkable proficiency in rapidly learning and retaining knowledge across a range of spatial environments. Furthermore, it facilitates an impressive transfer to unfamiliar settings. This capability for quick learning, generalization, and seamless adaptation represents a significant advancement in addressing complex cognitive tasks, which often pose challenges to conventional machine learning methods but are effortlessly handled by biological agents.

Experimental results illuminate not only the successful application of structured neural models to complex real-world tasks but also the potential limitations of traditional deep learning methodologies. These methods have historically grappled with issues such as rapid learning, generalization, and the avoidance of catastrophic forgetting. Our model deftly navigates these hurdles, underscoring the potential benefits of incorporating inductive biases into neural models.

Our findings carry implications for both artificial intelligence research and neuroscience. They suggest a promising role for structured neural models, inspired by architectures found in the brain, in tackling complex tasks, thereby pushing the boundaries of what artificial intelligence systems can currently achieve. Given these encouraging results, we believe that continued exploration and development of structured neural network models with low-dimensional rigid components may herald significant advancements in the field. Looking ahead, we consider it an exciting direction to explore how our proposed model could be further optimized or adapted to other, more general continual-learning tasks. Additionally, assessing its scalability and performance in even more complex, dynamic environments will be a valuable direction for future work.

Code

Our code is available at: https://github.com/raymondw2/seq-wm

Acknowledgements

IRF acknowledges funding from the NSF-CISE program, the ONR, the Simons Foundation through the SCGB, and the McGovern Institute at MIT.

Impact Statement

Although there is no distinct negative societal impact from the particular work in this paper, similar general cautionary notes apply as in research on continual learning methods and the development of AI.

References

Aimone, J. B., Deng, W., and Gage, F. H. Adult neurogenesis: integrating theories and separating functions. *Trends in cognitive sciences*, 14(7):325–337, 2010.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. *NeurIPS*, 2019.

Boschini, M., Bonicelli, L., Buzzega, P., Porrello, A., and Calderara, S. Class-incremental continual learning into the extended der-verse. *TPAMI*, 2022.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark experience for general continual learning: a strong, simple baseline. *NeurIPS*, 2020.

Caruana, R. Multitask learning. *Machine learning*, 28: 41–75, 1997.

Chandra, S., Sharma, S., Chaudhuri, R., and Fiete, I. *High-capacity flexible hippocampal associative and episodic memory enabled by prestructured "spatial" representations*. November 2023. doi: 10.1101/2023.11.28.568960.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-gem. In *ICLR*, 2019.

Cheung, B., Terekhov, A., Chen, Y., Agrawal, P., and Olshausen, B. Superposition of many models into one. *NeurIPS*, 2019.

Colgin, L. L., Moser, E. I., and Moser, M.-B. Understanding memory through hippocampal remapping. *Trends in neurosciences*, 31(9):469–477, 2008.

Douillard, A., Cord, M., Ollion, C., Robert, T., and Valle, E. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, 2020.

Fyhn, M., Hafting, T., Treves, A., Moser, M.-B., and Moser, E. I. Hippocampal remapping and grid realignment in entorhinal cortex. *Nature*, 446(7132):190–194, 2007.

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

Kang, M., Park, J., and Han, B. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *CVPR*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017.

Li, Z. and Hoiem, D. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. 1989.

Morris, R. G. Spatial localization does not require the presence of local cues. *Learning and motivation*, 12(2): 239–260, 1981.

O'Keefe, J. and Dostrovsky, J. The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 1971.

Personnaz, L., Guyon, I., and Dreyfus, G. Information storage and retrieval in spin-glass like neural networks. *Journal de Physique Lettres*, 46(8):359–365, 1985.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

Redish, A. D. and Touretzky, D. S. The role of the hippocampus in solving the morris water maze. *Neural computation*, 10(1):73–111, 1998.

Robins, A. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. *NeurIPS*, 2019.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Tapson, J. and van Schaik, A. Learning the pseudoinverse solution to network weights. *Neural Networks*, 45:94–100, 2013.

Thrun, S. and Pratt, L. Learning to learn: Introduction and overview. *Learning to learn*, pp. 3–17, 1998.

Van de Ven, G. M., Siegelmann, H. T., and Tolias, A. S. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):4069, 2020.

Vorhees, C. V. and Williams, M. T. Morris water maze: procedures for assessing spatial and related forms of learning and memory. *Nature protocols*, 1(2):848–858, 2006.

Wang, F.-Y., Zhou, D.-W., Ye, H.-J., and Zhan, D.-C. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, 2022.

Yan, S., Xie, J., and He, X. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, 2021.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong learning with dynamically expandable networks. *ICLR*, 2018.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *ICML*, 2017.

Zhou, D.-W., Wang, Q.-W., Qi, Z.-H., Ye, H.-J., Zhan, D.-C., and Liu, Z. Deep class-incremental learning: A survey. *arXiv preprint arXiv:2302.03648*, 2023.

A. Appendix

A.1. Dynamic Sequential Water Maze

In this section, we extend our original task, developing a more complex paradigm called the Dynamic Sequential Water Maze (dsWM). In the previous model, an agent was positioned within five distinct environments, each containing a unique goal location. However, the increased complexity of dsWM necessitates a more advanced set of cognitive capabilities from the agent.

In this version of the task, the agent must now retain the goal position for each environment while additionally *adapting* to altered goal positions within a single environment. To elaborate, the agent is initially trained in one environment, with a fixed goal position. Following this training period, the goal position is changed twice, yet the agent's policy is not trained further. The agent is then tested a further 800 times each in two different goal positions. This procedure is subsequently repeated in four additional unique environments, without the initial training period.

In our primary study, we compared our approach to several baseline models. These baseline models involved training the policy in one environment with a fixed goal position, followed by the relocation of the goal position within the same environment for testing. The results demonstrated that our method was unique in its ability to generalize without further training, while the baseline methods exhibited poor performance.

The Dynamic Sequential Morris Water Maze represents an extension of this original work, offering a more complex task and demanding greater cognitive adaptability from the agent. This enhanced task complexity will allow us to analyze the capability of our method further. Figure 5 shows that our vHSN is robust against all inter- and intra-environment changes.

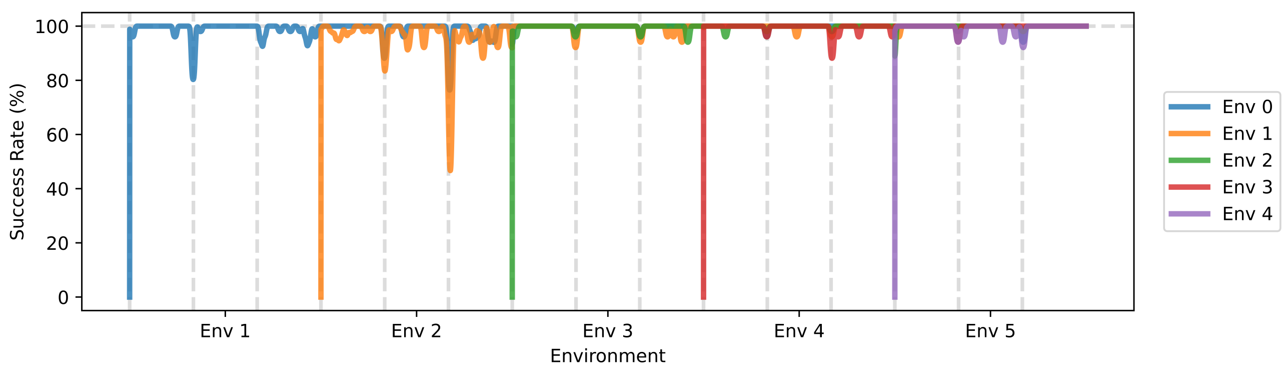

Figure 5. Our method shows robustness against all inter- and intra- environment changes. The average success rate of each environment while training on the following environments of our final model. Each dotted line indicates a goal position change. We use a moving average of 25 points and gaussian smoothing with $\sigma = 10$.

A.2. Replay Buffer

In order to assess the sensitivity of our baseline replay buffer, we conducted tests using a variety of replay buffer sizes, mirroring our original experimental setup. Initially, the network was trained in one environment and fine-tuned using 100 randomly sampled data points from the replay buffer. Subsequent testing was performed on all previously trained environments. The data obtained from these tests reveals no correlation between replay buffer size and performance in the latter environments, suggesting catastrophic forgetting.

A.3. Large Morris Water Maze

To assess the adaptability of our system, we doubled the size of our Morris Water Maze and observed a minor performance degradation (shown in Table 4). This test was conducted using the same number of time steps as in the smaller maze configuration. Despite the reduction in search time relative to the increased area, the system still maintained a high level of performance.

Table 4. The average success rate (%) of each environment after training all environments for two different sized sWM mazes.

Training Scheme	Average	Env 1	Env 2	Env 3	Env 4	Env 5
vHSN (15 x 15)	**99.2**	**99.2**	**99.5**	**99.0**	**99.5**	**98.8**
vHSN (30 x 30)	93.1	94.5	94.25	87.25	95.19	94.4

A.4. More Baselines

To benchmark against contemporary state-of-the-art methods, we selected four additional techniques: Dark Experience Replay (Buzzega et al., 2020), Dark Experience Replay ++ (Buzzega et al., 2020), A-GEM (Chaudhry et al., 2019), and Experience Replay (Rolnick et al., 2019).

In these benchmarks, we conducted tests over 100 epochs. For each epoch, the network is trained on 200 trajectories. Each trajectory was limited to a maximum of 100 time steps, after which we deemed it a timeout. Success was defined as the agent locating the goal within these 100 time steps. In methods utilizing a buffer, we set its capacity to 200. For the DER++ algorithm, we adhered to the optimal parameters recommended in the paper: α and β both set at 0.5. All tested methods employed Cross Entropy loss and a learning rate of 0.001. Post-training in all five environments, we assessed the accuracy of each on the previous environments. As a point of comparison, we introduce our method wherein the action-selection network is trained exclusively in the first environment and then subjected to zero-shot testing in the remaining environments.

A.5. Architecture Details

For our spatially invariant displacement network, we use a convolutional neural network (CNN) encoder (details in Table 5) for each of the three grid modules. The inputs to the encoder are the grid codes for the current location and the goal location. We use circular kernels. Then, we concatenate the three grid modules and pass into a feedforward encoder (details in Table 6) that outputs the displacement.

Type	**Input Channels/Features**	**Output Channels/Features**	**Kernel Size**	**Stride**
Conv2d	2	16	3	1
ReLU	-	-	-	-
Conv2d	16	32	3	1
ReLU	-	-	-	-
Conv2d	32	64	3	1
ReLU	-	-	-	-
AdaptiveAvgPool2d	-	-	-	-

Table 5. An architecture of a single CNN encoder for one grid module.

Type	**Input Channels/Features**	**Output Channels/Features**
Linear	192	132
ReLU	-	-
Linear	132	64
ReLU	-	-
Linear	64	64
ReLU	-	-
Linear	64	32
ReLU	-	-
Linear	32	4

Table 6. An architecture of a deep neural network that combines the displacement for the three grid modules.

A.6. Computing Infrastructure

We conducted our experiments on a high-performance computing system. The system was equipped with an AMD EPYC 7713 64-Core Processor, 32 GB of RAM and one Nvidia RTX 2080 Ti GPU.

A.7. Algorithm Pseudocode

Algorithm 1 Pseudocode for vHSN

```
agent = Agent() {Initalize agent }
attention = Attention() {Initialize attention block}
vector_hash = Vector-HaSH($\Lambda$, $N_{\text{place cells}}$) {Initialize Vector HaSH; $\Lambda$: set of grid periods}
all_obs = empty set() {Empty set to store observations}
goal_states = empty list() {Initialize goal as null}
envs = $[env_1, \ldots, env_n]$ {Initialize multiple environments}
for env in envs do
  for trajectory in env do
    if observation is associated with grid cell then
      found_where_i_am = true
    else
      found_where_i_am = false
    end if
    for step in trajectory do
      vHSN-update(env, agent, vector_hash, all_obs, found_where_i_am, goal_states)
    end for
  end for
end for
```

Algorithm 2 vHSN update

```
Input: env, agent, vector_hash, all_obs, found_where_i_am
obs = env.sensory_input {Get the current sensory input}
if First trial and not found_where_i_am then
    found_where_i_am = true
    vector_hash.remap_grid(obs) {Remap obs to a random set of grid states}
    grid_state = vector_hash.get_grid_activations()
end if
if found_where_i_am then
    vector_hash.update_weights(obs) {Associate observation with the current grid state}
    all_obs.add(obs)
    if goal is found then
        displacement = fixedCNN(goal_state, grid_state) {Calculate displacement}
        if in the first env then
            action = attention.associate(displacement, ground_truth_action) {Learn Association between displacement and
            action}
        else
            action = attention.retrieve(displacement)
        end if
    end if
else
    action = random_action() {Randomly wander until finding the goal and location}
end if
agent.step(action) {Agent takes a step in the env}
vector_hash.update_grid(action) {Update the grid state based on the action}
if env.reached_goal() then
    goal_state = grid_state {Update the goal if the agent reached it}
    goal_states.append(goal_state)
    break
end if
```

Algorithm 3 Vector-HaSH

```
$\text{self}.\mathbf{W}_{GP} = \frac{1}{|\mathbf{N}|}\sum_{\mu=1}^{\mu=N} \mathbf{g} \cdot (\text{sign}(\mathbf{W}_{PG} \cdot \mathbf{g}))^T$ {Initialize random connections between grid cells and place cells}
function remap_grid
    Input: obs
    $\text{self}.\mathbf{W}_{SP} = obs \cdot \mathbf{P}^\dagger$ {Hebbian learning to learn weights between observation and place cells}
    $\text{self}.\mathbf{W}_{PS} = \mathbf{P} \cdot obs^\dagger$
end function
function get_grid_activations
    $\mathbf{p}_t = \text{sign}(\text{self}.\mathbf{W}_{PS} \cdot \text{obs})$ {Get place activations from current observation}
    $\mathbf{g}_t = \text{CAN}(\text{self}.\mathbf{W}_{GP} \cdot \mathbf{p}_t)$ {CAN is a continuous attractor network}
    return $g_t$ {Return grid activations}
end function
function update_grid
    Input: action
    $g_{t+1} = A_{action}(gt)$ {$A_{action}$ shifts grid activations in the direction of movement}
end function
```