Multi-Environment MiniGrid World Models

Riju Mukherjee[0009−0009−0727−8164], Kurt Driessens[0000−0001−7871−2495], and Dennis J.N.J. Soemers[0000−0003−3241−8957]

Department of Advanced Computing Sciences, Maastricht University
r.mukherjee@student.maastrichtuniversity.nl
{kurt.driessens, dennis.soemers}@maastrichtuniversity.nl

Abstract. *World Models* in Reinforcement Learning agents support the construction of an internal representation of the external environment by compressing sensory experiences into a form suitable for reasoning, planning, and guiding behavior. They are inspired by the hippocampal formation in the limbic system of the mammalian brain, which facilitates spatial navigation, abstract problem-solving, generalization of knowledge, and transfer of learned skills across a wide range of contexts. In this paper, we consider two different world model architectures in the reinforcement learning setting: one using a *stochastic transformer*, and one using the hippocampus-inspired *TEM transformer*. We investigate the extent to which agents equipped with such world models can be effectively trained across a small set of diverse environments, and how well they transfer and generalize between them. Our experiments demonstrate early but promising signs that multi-environment agents can not only solve multiple tasks with shared parameters but also address the spatial invariance problem in a highly sample-efficient manner.

Keywords: World models · Generalization · Reinforcement learning.

1 Introduction

One of the cornerstones of human intelligence is our ability to generalize knowledge and transfer learned skills across different contexts and environments. These abilities allow humans to efficiently adapt in novel or semi-novel situations using their past experiences as a reference. The utility of such generalization and transfer can be observed in diverse daily-life scenarios, which humans perform with extreme efficiency and dexterity.

For example, supermarket chains usually follow a consistent pattern of the floor plan with the arrangements of different product categories across their different stores. However, each individual store often consists of spatial changes (e.g., the relative orientation of the entrance door, a vertically or horizontally mirrored floor plan, or a difference in the relative sizes of product sections). It takes relatively little effort for most humans to understand the pattern and adapt to a new store, considering they are already familiar with the pattern from their past shopping experiences in other stores of the same chain. Moreover, humans can navigate in new and unseen places, transfer motor skills, and

solve complex problems, because certain aspects of the world remain unchanged (e.g., gravitational force, the physics of opening doors, or the motor dynamics of climbing stairs). These universal regularities provide a stable foundation upon which learning in new contexts can be built. Whereas, shifts in subtle or more significant ways of more abstract dimensions, such as cultural norms and social etiquette, can be quickly recognized and, through adaptation, incorporated into human behavior [1].

One explanation of these abilities lies in our capacity to differentiate, memorize, and reuse underlying spatial and cognitive patterns from environment-specific sensory stimuli by storing knowledge of the world in a structured, coherent framework called *cognitive maps* [43]. The biological manifestation of cognitive maps has been associated with the hippocampal formation in the mammalian brain [35,18]. For spatial tasks, a variety of cell types, like the hippocampal place cells, landmark cells, and entorhinal grid cells, tune their firing pattern to solve the localization and mapping problem. However, during evolution, these mechanisms were repurposed for more generalized and non-spatial cognition and learning. The hippocampal formation became crucial for the organization and generalization of knowledge to novel experiences. Moreover, the interpretation of a cognitive map as a connected graph bridges the two domains of spatial and non-spatial cognition. In simpler terms, to the hippocampus, locomotion is equivalent to movement in spatial maps, whereas thinking or cognition is the ability to move in cognitive maps. For example, in mathematics, proving a new hypothesis is equivalent to creating or traversing along a cognitive graph where the starting or ending node is often an already established truth. Thus, humans can generalize by effectively reusing existing cognitive and spatial maps.

In contrast, generalization and adaptation remain challenging in reinforcement learning (RL) [44,51,50], significantly limiting the efficiency and real-world applicability of these systems. Despite the advent of deep learning-based RL approaches that promise scalable learning across tasks, agents are often still trained extensively on each environment before achieving good performance. *World Models* [13,15,14,17,52] could solve this problem in RL by building a compact representation of the environment dynamics and training the agent within this latent space, making the training process sampling and computation efficient. However, their impact on generalization and transfer is not tested.

Inspired by the roles that cognitive maps and the hippocampus appear to play in generalization and transfer learning in mammals, we consider two different world model architectures in the reinforcement learning setting: one using a *stochastic transformer* [52], and one using the hippocampus-inspired *TEM transformer* [45]. We present an exploratory study investigating the extent to which agents using such world models can effectively be trained in a small selection of multiple environments simultaneously, and transfer and generalize between them. Our experiments indicate that agents trained simultaneously on a handful of MiniGrid environments [5] can not only solve multiple tasks with shared parameters, but also address the spatial invariance problem in a highly sample-efficient manner.

2 Related Work

A common way to induce and benchmark generalization in RL is to train an agent on a wide array of variations of an environment (e.g., different levels of the same video game), and test it on yet another collection of variations [48,26,11,4,9,10,19,33,8,7,41,31]. However, such work typically relies on getting exposed to sufficient variety in environments at training time for generalization, more so than leveraging world models, specific architectures, or other inductive biases for generalization. Other multi-environment RL approaches rely on additional inputs to specify goals or otherwise enable disambiguation between environments. These inputs may take the form of, for example, numeric parameters [9], language-based instructions [30,2,32,34,40,47,28,12,27,20], or expert demonstrations [24,36,37]. Recent multi-environment world models are studied predominantly in large-scale settings (e.g., many dozens of environments and internet-scale data) [38,49].

3 World Model Architectures

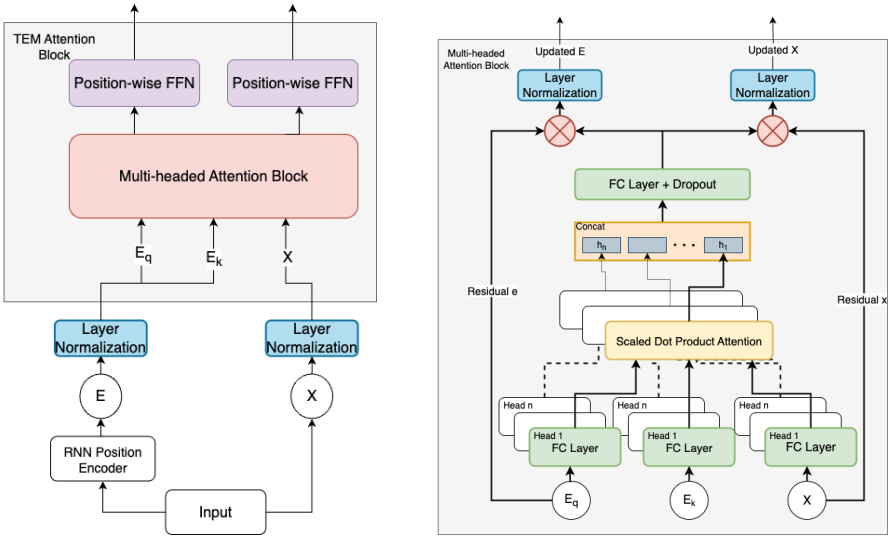
This section provides details on the world model architectures considered in this paper. We take a *stochastic transformer-based world model* [52] as a baseline, and introduce a variant with a *TEM transformer* [45] as an alternative.

3.1 Stochastic Transformer-based World Model (STORM) Architecture

The STORM architecture considered as a baseline in this paper closely follows the original [52]. It converts raw pixel-based observations o_t into stochastic categorical distributions $\mathcal{Z}_t$ using a Variational Auto-Encoder (VAE) [22]. Convolutional Neural Networks [23] are used as the main building blocks of the encoder and decoder of the VAE. Unlike standard VAEs, here $\mathcal{Z}_t$ is a categorical distribution comprising 32 categories, each with 32 classes. Additionally, a sample z_t is drawn from the latent variable $\mathcal{Z}_t$, and the straight-through gradient trick [3] is used to preserve the gradients for back propagation. Latent states z_t are combined with actions a_t into embedding tokens e_t, which in turn are passed to a transformer-based sequence model. The sequence model generates hidden states h_t, which are used for predictions of future latent states $\hat{Z}_{t+1}$, reward predictions $\hat{r}_t$, and predictions of whether or not the episode continues $\hat{c}_t$.

3.2 Tolman-Eichenbaum Machine (TEM) Transformer in World Model

The TEM transformer is a hippocampus-inspired transformer variant designed to model structured world knowledge through learned latent positions and associated sensory bindings [45]. Its core design draws on the Tolman-Eichenbaum



(a) TEM attention block with dual input and output stream: x_t and e_t (derived using the RNN position encoder). It uses the multi-headed attention block.

(b) The modified architecture of the TEM multi-headed attention block

Fig. 1: The architecture of the TEM transformer.

Machine (TEM) [46] principles. However, in the original paper, the TEM transformer was designed and tested for the prediction of spatial stimuli in a supervised learning setup; we customized and modified it to fit in as a sequence model of the world model architecture as shown in Fig. 1.

Whereas standard transformers process a single sequence, the TEM transformer processes two separate streams of embeddings:

- e_t: position (or latent spatial code) stream
- x_t: stimulus (observation) stream

Each attention block operates jointly on these streams via a modified multi-headed attention mechanism. Queries and keys are computed from the position stream e_t, while values are derived from the stimulus stream x_t. The attention computation is defined as:

$$\text{Attention}(X, E) = \text{softmax}\left(\frac{e_t E^T}{\sqrt{d_k}}\right) X \tag{1}$$

Here E, X are matrices with each row representing one timestep of e_t, x_t respectively, and d_k is the number of dimensions of e_t. The outputs are added to both

streams and normalized:

$$x'_t = \text{LayerNorm}(x_t + \text{Attention}(x_t, e_t)) \tag{2}$$
$$e'_t = \text{LayerNorm}(e_t + \text{Attention}(x_t, e_t)) \tag{3}$$

These updated streams are then passed through independent position-wise feed-forward layers. The dual residual pathways ensure that both spatial encoding and stimulus association evolve in tandem.

The original TEM transformer [45] uses a simple recurrent update rule for position encodings: $e_{t+1} = \sigma(e_t W_a)$, where W_a is a learnable action-dependent matrix and $\sigma(\cdot)$ is a non-linear activation function. Our implementation replaces this update with a Gated Recurrent Unit (GRU) [6]:

$$e_{t+1} = \text{GRU}(e_t, a_t). \tag{4}$$

The key difference is that the original update is a lightweight linear transformation with a nonlinearity, while the GRU introduces gating mechanisms. Theoretically, this should allow the position encoding to selectively retain, reset, or update information over time, leading to a more expressive and stable representation. This enables trajectory-aware encoding of spatial structure, resembling hippocampal path integration and grid cell formation. As the agent acts in the environment, the position representation evolves to reflect new positions, potentially supporting generalization and memory-based reasoning.

4 Experiments

This section describes the setup of our experiments, presents the results, and provides a discussion of them. The core aim of our experiments is to explore and investigate the extent to which generalization and transfer may take place in small-scale multi-environment training setups with world models.

4.1 Setup

Environments. For our experiments, we selected simple goal-oriented 2D grid-world environments from *MiniGrid* [5]. The agent in these environments is a red triangular object with a discrete action space consisting of seven possible actions. The `left` and `right` actions allow the agent to rotate its orientation, while the `forward` action moves the agent one step in the direction it is currently facing. The remaining actions—`pickup`, `drop`, `toggle`, and `done`—are defined within the environment but remain unused in the current experimental setup due to the choice of environments. The tasks involve solving different maze maps and interacting with various objects such as doors, keys, or boxes. The reward structure is discrete, designed to incentivize task completion in the minimum number of steps. Upon achievement of the goal, the agent receives a reward $R_{\text{success}} = 1 - 0.9 \cdot \left(\frac{step\_count}{max\_steps}\right)$, where step_count denotes the number of steps

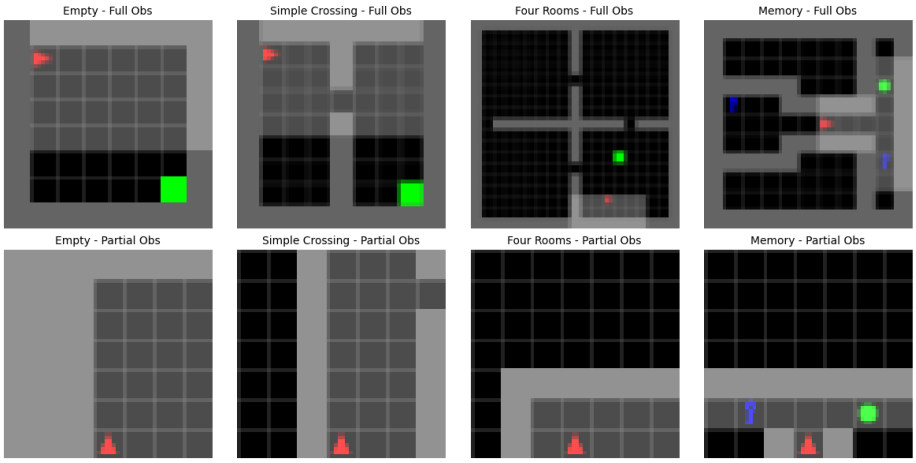


Fig. 2: Sample observations of the grid-world environments: *Empty, Simple Crossing, Four Rooms, and Memory*. The top row shows the fully observable variants, and the bottom row shows the partially observable variants (different samples).

taken to reach the goal, and max_steps is the maximum allowed number of steps per episode. Failure to complete the task within the allotted steps results in a reward of 0. We selected four environments, each with two variants: fully and partially observable. Fig. 2 shows random observations of the two variants of each of the four environments.

- **Empty**: It is the simplest of the four environments where the objective of the agent is to reach the goal: the green square. Upon reaching that, the agent gets a reward.
- **Simple Crossing**: Similar to the Empty environment, the goal is to reach the cell with a green square, but there are obstacles: walls that the agent can only cross through a specific opening. The position of the wall and the opening is random in each episode.
- **Four Rooms**: The agent must navigate in a maze composed of four rooms interconnected by four gaps in the walls. To obtain a reward, the agent must reach the green goal square. Both the agent and the goal square are randomly placed in any of the four rooms. It is an extension of *Simple Crossing* that requires Hierarchical RL capabilities to solve [42].
- **Memory**: This environment is a memory test. The agent starts in a small room where it sees an object. It then has to go through a narrow hallway, which ends in a split. At each end of the split, there is an object, one of which is the same as the object in the starting room. The agent has to remember the initial object and go to the matching object at the split.

The partially observable variants are the egocentric views of the fully observable environments. In these cases, the agent object (red triangle) remains fixed

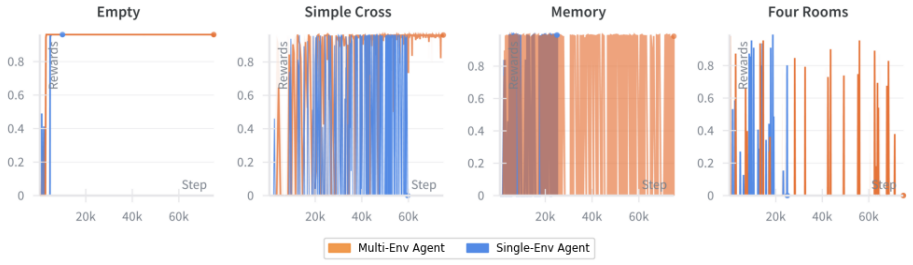
in its position at the bottom of the frames, whereas the surroundings change based on what the agent is supposed to observe. The limiting reach of the view significantly alters the nature of the environments. For example, the partially-observable *Memory* environment requires the agent to memorize the initially visited object (blue key or green ball) and reach a similar object as the goal. However, under full observation, the need to remember the initial object becomes superfluous, as the whole environment is always accessible to our AC-agent.

Hyperparameters. Across all experiments, certain elements of the core architecture and hyperparameters of the world model and the actor-critic agent were kept consistent to ensure fair comparisons. Any hyperparameters, choice of loss functions, training setup, and so on that are not specified otherwise are kept the same as in the original STORM publication [52]. The input to the world model comprises environment frames as RGB images of size $64 \times 64 \times 3$. The encoder is implemented as a convolutional backbone with batch normalization, starting with a stem layer of 32 channels and progressively downsampling until the spatial resolution reaches 4×4. The final encoded feature dimension is given by $C_{\text{last}} \times 4 \times 4$. The latent stochastic state has dimension $K \times K$ with $K = 32$, flattened to 1024 features.

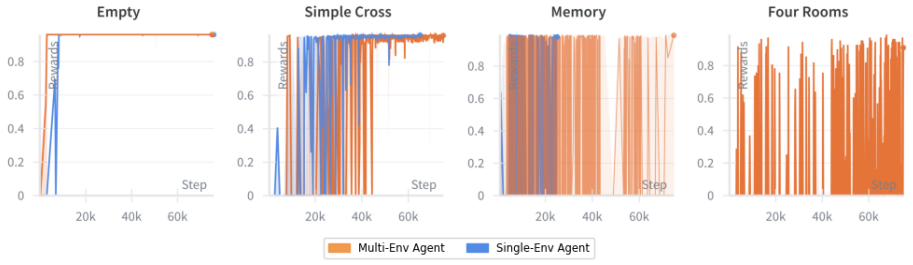
Both the baseline *stochastic transformers* and the *TEM transformer* are initiated with two transformer blocks with 512 hidden dimensions, eight attention heads, and a dropout rate of 0.1. Three prediction heads are employed: the distribution head maps the encoder output to posterior logits and the transformer states to prior logits; the reward predictor is a two-layer MLP with hidden dimension 512, LayerNorm, and ReLU activations, outputting a discrete 255-class symlog-twohot reward encoding [17]; and the continuation predictor, implemented as a similar MLP, predicts the scalar termination probability. The decoder is a de-convolutional network reconstructing the 64×64 RGB observation from the latent state. The world model is trained with Adam optimizer [21] at a learning rate of 1×10^{-4}, gradient clipping at 1000, mixed-precision training (AMP) enabled, and a free-bits threshold of 1 for KL regularization.

The actor-critic operates on the concatenation of latent samples and transformer hidden states, producing policy logits and value estimates. The input dimension of the actor-critic is $1024 + 512 = 1536$. The actor network is a two-layer MLP with hidden size 512, LayerNorm, and ReLU activations, mapping to discrete action logits by first passing the logits to a categorical distribution and then sampling from it. The critic shares the same backbone but outputs a 255-dimensional symlog-twohot value representation, with a slow critic maintained as an exponential moving average copy updated with a decay rate of 0.98. Training is performed with Adam at a learning rate of 3×10^{-5}, $\epsilon = 1 \times 10^{-5}$, gradient clipping at 1000.0, and AMP enabled. The discount factor is $\gamma = 0.985$, the GAE [39] parameter is $\lambda = 0.95$, and an entropy regularization coefficient of 3×10^{-4} is applied.

4.2 Results



(a)



(b)

Fig. 3: Comparison of performance of baseline world model across (a) fully and (b) partially observable environments between agents trained only on single environments, with agents trained on all environments simultaneously. The plots show the episodic returns as a function of environment steps. The cumulative reward is recorded only at the termination of each episode, after which a new episode begins while the environment step count continues to increase.

Multi-Environment and Single-Environment Training. In this experiment, the same model architecture and hyperparameters are used to train the agent (world model and actor-critic) across all four environments simultaneously. During the initial 4000 steps, vectorized environments were sampled asynchronously, and the resulting transitions `[observation, action, reward, info]` were stored in a replay buffer. Subsequently, for the remaining training steps, batches were drawn from the replay buffer to train the world model and actor-critic agent. Each batch consisted of ordered sequences of a fixed length, sampled from all environments, resulting in data of shape `[batch_size, imagination_length, H]`, where `batch_size` and `imagination_length` were set to 128 and 32, respectively, and `H` depends on the feature dimensionality of each element in the transitions data. Through this process, the shared parameters of the agent were optimized to maximize cumulative rewards across all environments concurrently.

We compare the performance of the multi-environment agent (MEA) to that of a set of single-environment agents (SEAs), each trained independently on a single environment from scratch without any parameter sharing. In both cases,

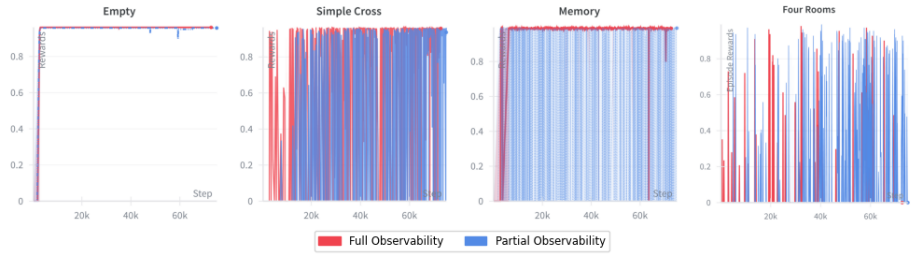


Fig. 4: Multi-environment agent (MEA) training with TEM transformer-based world model in different environments. The plots show the episodic returns as a function of environment steps. The cumulative reward is recorded only at the termination of each episode, after which a new episode begins while the environment step count continues to increase.

the *stochastic transformer* was utilized as the sequence model in the world model architecture. The comparative results are presented in Figure 3, where we plot the episodic returns as a function of the number of environment interactions. The *Step* count increases monotonically as training progresses and the agent transitions across new episodes. A very high episodic return indicates that the agent could solve the environment, and the more optimal the solution, the closer the return is to 1. After each environment step, the collected transition was stored in the replay buffer, from which updates to both the world model and the actor-critic were subsequently performed. Another MEA experiment was performed with *TEM-transformer* as the sequence model in the world model, keeping other settings unchanged. The results are shown in Figure 4.

Latent Representations of Environments. To investigate how the multi-environment agent (MEA) represents different environments, we analyze the latent representations learned by the world model. Once the agent is trained, in evaluation, we roll out a fixed number of steps (16 per environment) and capture the hidden states of the transformer in the world model. Since this hidden representation serves as the shared input to the reward head, continuation head, and actor-critic module, it constitutes a common latent space influencing multiple components. These hidden states are then averaged across the time dimension, resulting in a single aggregated vector for each environment.

To visualize these representations, we apply t-SNE [29] to the aggregated environment vectors at different training checkpoints for both the fully and partially observable environments, as shown in Figure 5. Initially, the environment representations are close together; however, as training progresses, they increasingly get separated into distinct clusters. This trend is observed for both the baseline *stochastic transformer* and *TEM transformer* architectures. In the fully observable environments, the *Memory* environment consistently forms a cluster distant from the others, likely due to its visual and structural dissimilarity.

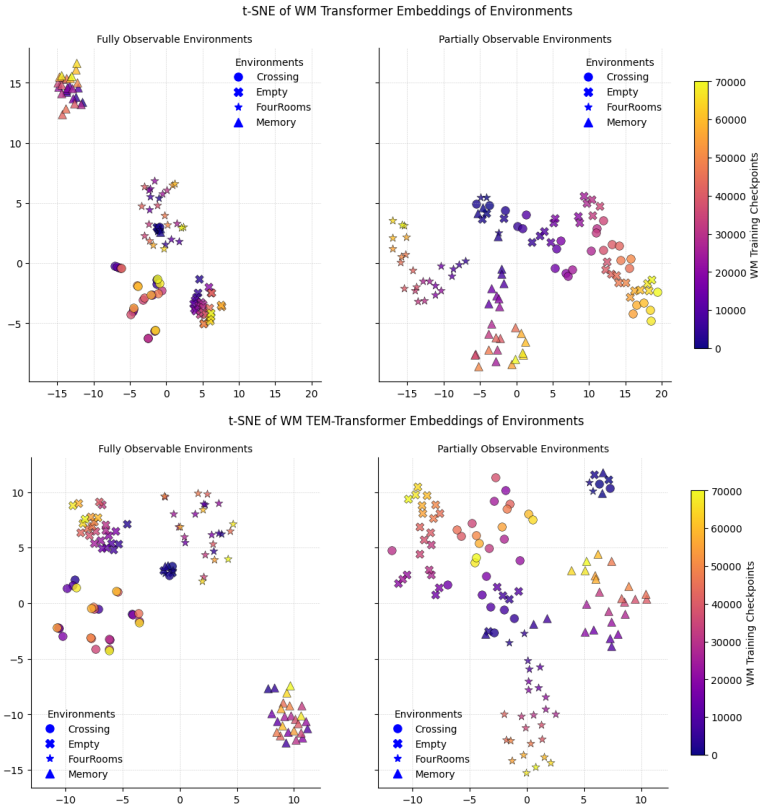


Fig. 5: T-SNE plot of the hidden embeddings of the world model transformers across various training checkpoints. Top: embeddings of the baseline stochastic transformer. Bottom: embeddings of the TEM transformer. The left column shows the fully observable environments, and the right column shows their partially observable counterparts.

For the partially observable settings, the latent clusters of *Empty* often appear close to those of *Simple Cross*, which can be attributed to the similarity in their observations in visual space when partial observability is enforced.

Spatial Invariance. Spatial invariance can be characterized as the notion of identifying similarities of different environments with spatial modifications like rotation, elongation, expansion, etc. To test the agent's robustness against such variations, we first fabricate some additional environments. We consider the fully-observable variant of the *Empty* environment and change the relative positions of the agent (red triangle) and the goal (green square) while keeping the head direction of the agent fixed. We create three such environments, referred to as *Empty-FullObs-Ort-1, Empty-FullObs-Ort-2, Empty-FullObs-Ort-3*, while the

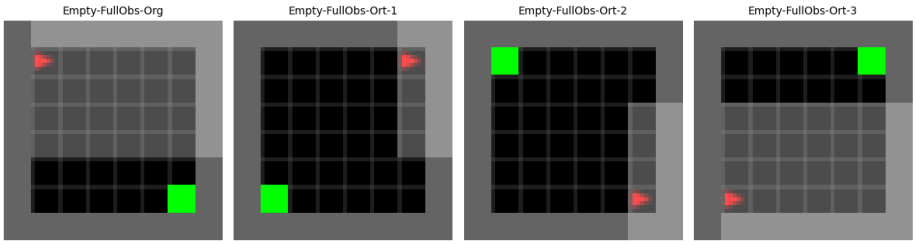


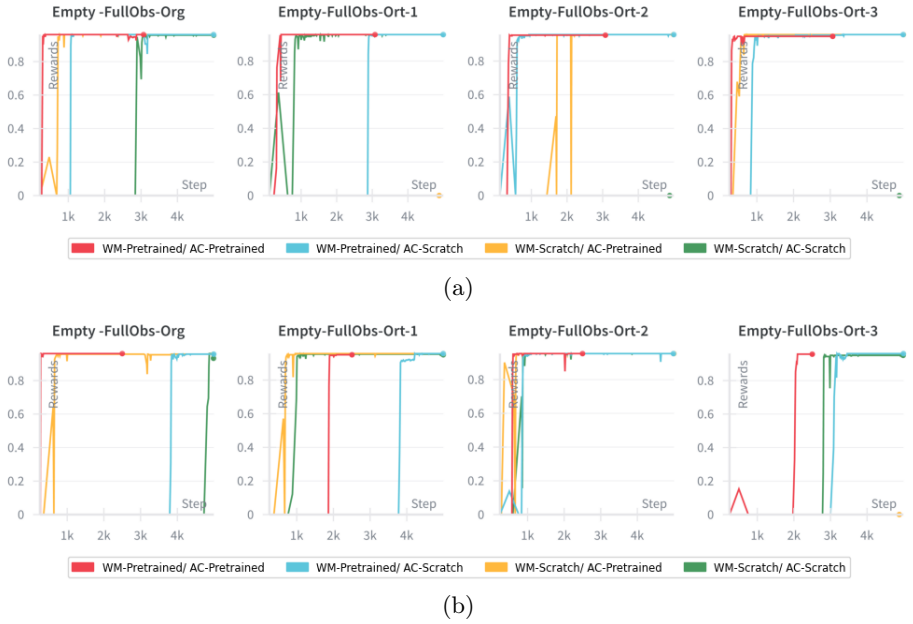
Fig. 6: Different variants of the fully observable *Empty* environment achieved by changing the relative position of the agent and the goal. These modified environments are: original *Empty*, *Empty-FullObs-Ort-1, Empty-FullObs-Ort-2, Empty-FullObs-Ort-3* from left to right

original is referred to as *Empty-Org* as shown in Fig. 6. Each of these environments has an effective change equivalent to a consecutive 90° clockwise rotation. The remaining dynamics of the environment and the reward structure remain the same.

To analyze the agent's few-shot adaptation ability to spatial variations described above, we take existing model parameters of the trained agents in the fully-observable multi-environment setting, and run them in these new environments. We try four different combinations based on whether the world model and the actor-critic are loaded from pretrained weights or trained from scratch. The results are displayed in Fig. 7. When both the world model and actor-critic are initiated from the pre-trained weights, the agent learns the optimal policy in less than 50 steps of training (the first 250 steps are used for buffer warm-up). This is faster than all the other three combinations, where at least one or both models are trained from scratch. Both the transformer variants show a similar trend, with one exception for *TEM transformer* in the *Empty-FullObs-Ort-1* environment.

4.3 Discussion

In the fully observable setting, the MEA successfully learned optimal policies in *Empty* and *Simple Cross*, as the episodic return is consistently high. In *Memory* and *Four Rooms*, the zigzag lines indicate oscillatory episodic return between very-high and very-low. This signifies that the agent could not consistently solve the environment. On the other hand, SEAs achieved optimal performance only in *Empty* (see Fig. 3). In partially observable environments, both MEA and SEAs reached optimal policies in *Empty* and *Simple Cross*. The *Four Rooms* environment was originally proposed as a benchmark for hierarchical RL approaches [42], which were not integrated in our work. Hence, integrating hierarchical RL capabilities inside the world model [16] would be a natural way to improve performance in this environment. We remark that, with sufficient transfer from other environments such as *Simple Crossing* (which ought to teach an agent



(a)

(b)

Fig. 7: Comparison of agent performance in semi-novel environments at different degrees of transfer. On the top (a), we observe results achieved with baseline-transformer, (b) shows the result achieved using *TEM transformer*. The plots show the episodic returns as a function of environment steps. The cumulative reward is recorded only at the termination of each episode, after which a new episode begins while the environment step count continues to increase.

how to navigate towards a green objective and cross doorways), this might not have been necessary, but we do not observe such an extent of generalization happening in our experiments.

The primary change in results after switching to the *TEM transformer* (see Fig. 4) is that the *Memory* environment appears to become solvable, but still only in the fully observable setting (in which there is no actual need for memorization). Further research (e.g., repetitions with more random seeds) is needed before this difference can be conclusively attributed to the modified transformer type.

The t-SNE embeddings of Fig. 5 show that networks trained on multiple environments simultaneously clearly become able to distinguish between them as learning progresses. On the one hand, this may be seen as a necessity for strong performance across different environments (as an agent that is confused as to which environment it is in cannot be expected to perform well). On the other hand, the clean separation of environments may be indicative of a lack of transfer between them.

The results of Fig. 7 demonstrate that, especially when both the world model and the actor-critic are transferred to the rotated versions of the *Empty* environment, training consistently accelerates in comparison to training from scratch. This is indicative of a helpful transfer.

5 Conclusion

Inspired by the role that world models play in the intelligence and generalization capabilities of humans, we described an exploratory study of the abilities of reinforcement learning agents to train simultaneously in a small selection of multiple different MiniGrid environments, and generalize and transfer between and from them. This is achieved with small-scale RL: agents are only trained on up to four different environments, from which their ability to generalize is assessed. We used STORM [52] as a state-of-the-art baseline world model, and also tested a variant of it in which we replaced its stochastic transformer with a hippocampus-inspired TEM-transformer [46]. Experimental results demonstrate an ability to learn effective policies in multiple distinct environments in a single shared network, with no need for explicit environment-specific context. Benefits of transfer and the ability to effectively recognise and distinguish between different environments are observed. However, consistent success in the two most complex of the four environments remains out of reach for the tested approaches.

Future work aimed at developing a more thorough understanding of generalization and transfer behaviors could consist of more extensive experiments with carefully designed variations of environments. For example, the colors of objects could be modified to evaluate the extent to which trained agents can or cannot generalize across environments based on the colors of goal or hazard objects. Furthermore, the array of environments that an agent is simultaneously trained on could be extended to include more environments that are conceptually "in between" current ones, possibly forming a bridge to enable smoother transfer between them. To improve the performance of agents in general, future work could consider algorithmic and architectural additions such as Mixture-of-Experts [25] or hierarchical RL support for world models [16].

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Anderson, M.L.: Neural reuse: A fundamental organizational principle of the brain. Behavioral and Brain Sciences **33**(4), 245–266 (2010)
2. Andreas, J., Klein, D., Levine, S.: Modular multitask reinforcement learning with policy sketches. In: Proceedings of the 34th International Conference on Machine Learning. vol. 70, pp. 166–175. PMLR (2017)
3. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013), introduces the straight-through estimator (STE)

4. Bou Ammar, H., Eaton, E., Ruvolo, P., Taylor, M.E.: Online multi-task learning for policy gradient methods. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 1206–1214 (2014)
5. Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P.S., Terry, J.: Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. CoRR **abs/2306.13831** (2023)
6. Cho, K., van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder–decoder approaches. In: Wu, D., Carpuat, M., Carreras, X., Vecchi, E.M. (eds.) Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation. pp. 103–111. Association for Computational Linguistics (2014)
7. Cobbe, K., Hesse, C., Hilton, J., Schulman, J.: Leveraging procedural generation to benchmark reinforcement learning. In: Daumé III, H., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 2048–2056 (2020)
8. Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 1282–1289. PMLR (2019)
9. Deisenroth, M.P., Englert, P., Peters, J., Fox, D.: Multi-task policy search for robotics. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 3876–3881 (2014)
10. Farebrother, J., Machado, M.C., Bowling, M.: Generalization and regularization in DQN. https://arxiv.org/abs/1810.00123 (2018)
11. Fernandéz, F., Veloso, M.: Learning domain structure through probabilistic policy reuse in reinforcement learning. Progress in AI **2**(1), 13–27 (2013)
12. Goyal, P., Niekum, S., Mooney, R.J.: PixL2R: Guiding reinforcement learning using natural language by mapping pixels to rewards. In: Proceedings of the 2020 Conference on Robot Learning. PMLR, vol. 155, pp. 485–497 (2021)
13. Ha, D., Schmidhuber, J.: World models. arXiv preprint arXiv:1803.10122 (2018)
14. Hafner, D., Lillicrap, T., Ba, J., Norouzi, M.: Dream to control: Learning behaviors by latent imagination. arXiv preprint arXiv:1912.01603 (2019)
15. Hafner, D., Lillicrap, T., Norouzi, M., Ba, J.: Learning latent dynamics for planning from pixels. In: International Conference on Machine Learning (ICML) (2019)
16. Hafner, D., Pasukonis, J., Ba, J.: The director: A latent agent for control. arXiv preprint arXiv:2303.04137 (2023)
17. Hafner, D., Pasukonis, J., Ba, J., Lillicrap, T.: Mastering diverse domains through world models. arXiv preprint arXiv:2301.04104 (2023)
18. Hafting, T., Fyhn, M., Molden, S., Moser, M.B., Moser, E.I.: Microstructure of a spatial map in the entorhinal cortex. Nature **436**(7052), 801–806 (2005)
19. Justesen, N., Torrado, R.R., Bontrager, P., Khalifa, A., Togelius, J., Risi, S.: Illuminating generalization in deep reinforcement learning through procedural level generation. In: NeurIPS 2018 Workshop on Deep Reinforcement Learning (2018)
20. Kharyal, C., Krishna Gottipati, S., Kumar Sinha, T., Das, S., Taylor, M.E.: GLIDE-RL: Grounded language instruction through DEmonstration in RL. https://arxiv.org/abs/2401.02991 (2024)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

22. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: International Conference on Learning Representations (ICLR) (2014), originally published as arXiv preprint arXiv:1312.6114, 2013
23. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
24. Lee, J.N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., Brunskill, E.: Supervised pretraining can learn in-context reinforcement learning. https://arxiv.org/abs/2306.14892 (2023)
25. Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., Chen, Z.: Gshard: Scaling giant models with conditional computation and automatic sharding. arXiv preprint arXiv:2006.16668 (2020)
26. Li, H., Liao, X., Carin, L.: Multi-task reinforcement learning in partially observable stochastic environments. Journal of Machine Learning Research **10**(40), 1131–1186 (2009)
27. Lifschitz, S., Paster, K., Chan, H., Ba, J., McIlraith, S.: Steve-1: A generative model for text-to-behavior in Minecraft. https://arxiv.org/abs/2306.00937 (2023)
28. Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., Rocktä"schel, T.: A survey of reinforcement learning informed by natural language. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 6309–6317 (2019)
29. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
30. Maclin, R., Shavlik, J.W.: Creating advice-taking reinforcement learners. Machine Learning **22**, 251–281 (1996)
31. Mediratta, I., You, Q., Jiang, M., Raileanu, R.: A study of generalization in offline reinforcement learning. In: 2024 International Conference on Learning Representations (2024)
32. Misra, D., Langford, J., Artzi, Y.: Mapping instructions and visual observations to actions with reinforcement learning. In: Palmer, M., Hwa, R., Riedel, S. (eds.) Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 1004–1015 (2017)
33. Nichol, A., Pfau, V., Hesse, C., Klimov, O., Schulman, J.: Gotta learn fast: A new benchmark for generalization in RL. https://arxiv.org/abs/1804.03720 (2018)
34. Oh, J., Singh, S., Lee, H., Kohli, P.: Zero-shot task generalization with multi-task deep reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning. pp. 2661–2670. PMLR (2017)
35. O'Keefe, J., Nadel, L.: The hippocampus as a cognitive map. Oxford University Press (1978)
36. Raparthy, S.C., Hambro, E., Kirk, R., Henaff, M., Raileanu, R.: Generalization to new sequential decision making tasks with in-context learning. https://arxiv.org/abs/2312.03801 (2023)
37. Reed, S., Żołna, K., Parisotto, E., Colmenarejo, S.G., Novikov, A., Barth-Maron, G., Giménez, M., Sulsky, Y., Kay, J., Springenberg, J.T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., de Freitas, N.: A generalist agent. Transactions on Machine Learning Research (2023)
38. Schubert, I., Zhang, J., Bruce, J., Bechtle, S., Parisotto, E., Riedmiller, M., Springenberg, J.T., Byravan, A., Hasenclever, L., Heess, N.: A generalist dynamics model for control. https://arxiv.org/abs/2305.10912 (2023)

39. Schulman, J., Moritz, P., Levine, S., Jordan, M.I., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. In: International Conference on Learning Representations (ICLR 2016) (2016)
40. Shu, T., Xiong, C., Socher, R.: Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In: International Conference on Learning Representations (2018)
41. Stone, A., Ramirez, O., Konolige, K., Jonschkowski, R.: The distracting control suite – a challenging benchmark for reinforcement learning from pixels. https://arxiv.org/abs/2101.02722 (2021)
42. Sutton, R.S., Precup, D., Singh, S.: Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial intelligence **112**(1-2), 181–211 (1999)
43. Tolman, E.C.: Cognitive maps in rats and men. Psychological Review **55**(4), 189–208 (1948)
44. Whiteson, S., Tanner, B., Taylor, M.E., Stone, P.: Protecting against evaluation overfitting in empirical reinforcement learning. In: 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). pp. 120–127 (2011)
45. Whittington, J.C.R., Warren, J., Behrens, T.E.J.: Relating transformers to models and neural representations of the hippocampal formation. In: International Conference on Learning Representations (ICLR) (2022)
46. Whittington, J.C., Muller, T.H., Mark, S., Chen, G., Barry, C., Burgess, N., Behrens, T.E.: The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. Cell **183**(5), 1249–1263 (2020)
47. Williams, E.C., Gopalan, N., Rhee, M., Tellex, S.: Learning to parse natural language to gronuded reward functions with weak supervision. In: 2018 IEEE International Conference on Robotics and Automation. pp. 4430–4436 (2018)
48. Wilson, A., Fern, A., Ray, S., Tadepalli, P.: Multi-task reinforcement learning: A hierarchical Bayesian approach. In: Proceedings of the 24th International Conference on Machine Learning. pp. 1015–1022 (2007)
49. Yin, S., Wu, J., Huang, S., Su, X., He, X., Hao, J., Long, M.: Trajectory world models for heterogeneous environments. In: Proceedings of the 42nd International Conference on Machine Learning (2025), accepted
50. Zhang, A., Ballas, N., Pineau, J.: A dissection of overfitting and generalization in continuous reinforcement learning. https://arxiv.org/abs/1806.07937 (2020)
51. Zhang, C., Vinyals, O., Munos, R., Bengio, S.: A study on overfitting in deep reinforcement learning. https://arxiv.org/abs/1804.06893 (2018)
52. Zhang, W., Wang, G., Sun, J., Yuan, Y., Huang, G.: Storm: Efficient stochastic transformer-based world models. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol. 36, pp. 27147–27166. Curran Associates, Inc. (2023)