Context-Switching and Adaptation: Brain-Inspired Mechanisms for Handling Environmental Changes

Eric Chalmers, Edgar Bermudez Contreras, Brandon Robertson, Artur Luczak, and Aaron Gruber
Canadian Centre for Behavioural Neuroscience
University of Lethbridge, Alberta, Canada
eric.chalmers@uleth.ca

Abstract— **Reinforcement learning (RL) allows an intelligent agent to learn optimal behavior as it interacts with its environment. Conventional model-based RL algorithms learn rapidly, but can be slow to adapt to sudden changes in the environment. Animals' brains, however, are thought to employ model-based RL mechanisms for learning, but are able to adapt to changes with relative ease. By employing "transfer learning", they can recycle previously learned information to solve new problems with minimal new learning. We developed two brain-inspired methods that can allow model-based RL to cope with changes to the underlying process being learned: hierarchical state abstraction, and context-switching. Hierarchical state abstraction allows a previously-learned model to be efficiently adapted for use in a new task, while context switching allows learned models to be saved and recalled at the appropriate times. We test these mechanisms using grid-world simulations in which the goal remains constant, but contingencies for reaching it frequently change. These mechanisms allow an agent to significantly outperform a conventional model-based RL algorithm in the task.**

Keywords—reinforcement learning; transfer learning; dynamic environment

I. Introduction

Transfer learning is an important component of general intelligence – it describes the ability of an agent to learn the solution to one problem, and then apply the acquired knowledge to solve other related problems. Thrun and Pratt describe transfer learning as "learning to learn". They explain that an artificial intelligence is said to *learn* if its performance at some task improves with experience, and that it is said to *learn to learn* if its performance at several tasks improves with experience *and* with the number of tasks [1]. The ability to store, retrieve, and modify previously learned knowledge is prerequisite for this kind of learning.

Reinforcement learning (RL) is a machine learning paradigm in which an intelligent agent may learn optimal behavior by interacting with its environment [2]. It effectively allows the agent to learn a policy (i.e. mappings from the agent's current state to an appropriate action) which maximizes its reward in some task, though conventional RL is perhaps best suited to learning a single task. RL assumes the agent is learning the solution to a Markov Decision Process (MDP). The MDP involves a set of states which the agent can occupy, and a set of actions available in each state. It assigns some probability *P(s'|s,a)* of transition between a state *s*, and a new state *s'*, given action *a*. It also includes a reward function *R(s,a,s')*, associating a scalar reward with each transition. The value of executing action *a* from state *s* is:

$$V(\mathrm{s},a) = E[\mathrm{reward} + \gamma \times \max_{a'} V(\mathrm{s}',a')] \quad (1)$$

where γ ϵ (0,1) is a discount factor that causes immediate rewards to be favored over future ones, and *a'* is an action executed from the new state *s'*. RL is the process of learning accurate action values for each state-action pair. This work focuses on model-based RL [3], in which the agent builds up internal models of *P(s'|s,a)* and *R(s,a,s')* based on its experiences while navigating the MDP. A value estimate can then be calculated:

$$\bar{V}(\mathrm{s},a) = \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma \max_{a'} \bar{V}(\mathrm{s}',a')\big) \quad (2)$$

RL is a prominent concept in both the artificial intelligence and neuroscience communities. In the brain, dopamine-driven learning mechanisms have the reward signaling properties needed to iteratively solve Eq. 2, and trial-and error learning by animals often has properties predicted by RL [4]. However, while animals seem quite good at transfer learning – learning to solve multiple problems and re-using previously learned information – model-based RL is not. This is because the agent's set of value estimates creates a value gradient, which pushes the agent in the most rewarding direction from a given state. This value gradient becomes invalid if the task changes, and it takes a considerable amount of time and experience for the agent to unlearn the previous gradient and form appropriate action values from the changing *P(s'|s,a)* and *R(s,a,s')* models. In the worst case, the most rewarding action in the *new* task is thought by the *old* model to be the least rewarding: in this case model-based RL may only discover the new reward contingencies by means of random exploration (see Fig. 1).

Since humans and animals can handle task changes with relative ease, the brain may implement model-based RL in a way that accommodates storage, reuse, and interchanging of previously learned information. The machine learning community has proposed various mechanisms of incorporating transfer learning in RL, many of which are reviewed by Taylor and Stone [5]. Features of the hippocampus – a brain structure associated with spatial representation and navigation – hint at

This work was funded in part by The Natural Sciences and Engineering Research Council of Canada (NSERC)

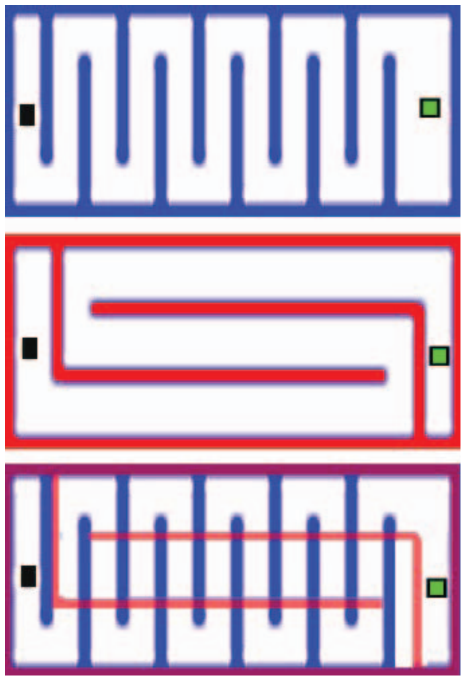
Fig. 1. Gridworld simulations in which the agent (marked in black) must reach a goal (marked in green). If a conventional model-based RL agent learns to solve a first maze (shown in blue), and is then switched to a second (shown in red), information about the first maze's boundaries persists in the agent's model of *P(s'|s,a)*. As a result, the value-calculation process comes to believe that there is no route to the goal. Only through random exploratory actions would the agent discover the new true transition probabilities.

two mechanisms in particular: hierarchical representation of the state space, and context switching.

A. *Spatial hierarchy in the brain*

Hierarchical abstraction is a useful tool for simplifying complex learning tasks, both for artificial agents and humans. The greater the level of abstraction, the simpler the task becomes. Previous work has incorporated hierarchical abstraction into reinforcement learning in various ways. The "Options" framework presented by Precup [6], [7] uses a hierarchy in the *action space*, grouping sequences of primitive actions into macro actions which can be executed atomically. Appropriate macro-actions can greatly simplify a learning problem: a goal which is only a few macro-actions away may have been many steps away in the original problem. Botvinick et al [8], [9] have discussed humans' use of this kind of abstraction – for example, in learning to grasp, lift, and tilt a jug of water as a single action. The MAXQ framework proposed by Dietterich [10] incorporated hierarchy in the *task space*. A complex task is decomposed into simpler subroutine-like tasks, which can be learned individually and re-used as appropriate throughout the task.

Hierarchical abstraction can also be applied in the *state space*. State-space abstraction has been incorporated into RL through elimination of state variables that are irrelevant for a current sub-task [11], [12], and by merging adjacent states into macro-states [13], [14]. This type of abstraction is, perhaps, less studied in RL than action and task-space abstraction, but it may be important in the brain. The hippocampus – a brain structure associated with spatial representation and navigation – contains neurons called "place cells" which fire only when the animal is located in a "place field" that corresponds to a particular region of an environment. Interestingly, as one moves along the dorsal-ventral axis of the hippocampus, the size of the place fields changes: some cells represent specific places, while others represent larger regions [15]. Thus, the hippocampus seems to employ a hierarchical representation of space.

Such a hierarchical representation of space allows efficient hierarchical planning, as in Botea et al. [16], and we propose that a hierarchical reinforcement learning and planning system can allow an agent to efficiently adapt to changes in its environment, especially in spatial navigation tasks, or tasks with a spatial component.

B. *Context-switching in the brain*

Place cells in the hippocampus display the interesting feature of "remapping", changing their firing characteristics in response to changes in the environment. Subtle changes to a familiar environment can alter the firing rate at which a particular place cell used to fire before the changes took place. These slight changes to the cognitive map are called "rate remapping". More dramatic environmental changes trigger "general remapping" of place cells. That is, place cells representing one set of locations in a particular environment may be recruited to represent a new set of locations in a second environment. If the animal is returned to the first environment, the cells return to the original configuration learned in that environment [17]. Thus, the brain can modify its internal representation of an environment to account for changes, store this representation, and retrieve it again the next time it encounters that environment.

It has been speculated that the hippocampus houses the "model" for a model-based reinforcement learning scheme [18]. If this is true, it seems the animal is effectively able to store and retrieve models as appropriate. It also alters models in response to subtle changes in the process being modelled, possibly by adapting a similar model stored previously. With the ability to store, adapt, and retrieve previously learned information, the agent could learn to perform a variety of tasks without encountering the problem illustrated in Fig. 1.

C. *Related contributions to transfer learning in RL*

Transfer of knowledge between tasks has been considered by previous RL research. For example, Redish et al. [20] used a simple form of context switching to model relapse after addiction recovery - manually supplying a "situation" stimulus which caused the agent to toggle between two separate learned behaviors. Lazaric [21] developed a system of storing the agent's individual experiences across many tasks. Experiences from a new task were used to retrieve similar prior ones to create a set of data on which batch learning was performed; ultimately allowing individual experiences to generalize across tasks. Mehta et al [22] considered the problem of differing reward functions across otherwise similar tasks. They combined task-space hierarchy with a method of matching current reward information with previously observed reward functions to achieve effective transfer learning. Atkeson and Santamaria [23]

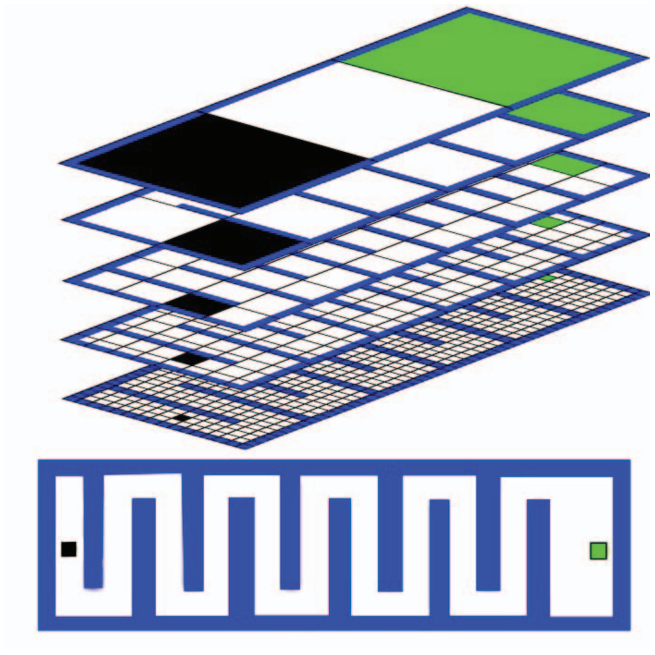

Fig. 2. Hierarchical representation of space in a simple gridworld task. Groups of adjacent states are aggregated into macro-states at progressively higher levels of abstraction.

considered the same problem of differing reward functions, demonstrating that a model of the transition function could be transferred from a source to a target task. Fernandez and Veloso [24] considered tasks in which only the goal state differs. They probabilistically selected between learned policies such that more useful policies were selected more often. The concepts proposed here complement this previous research.

D. *Contributions of this work*

Our work presented here builds upon previous work in machine learning by combining a hippocampus-inspired state-space hierarchy with a mechanism for storing and retrieving models based on recent experiences. The storage and retrieval of learned models allows multiple tasks to be learned separately, while the state-space hierarchy allows previously-learned models to be adapted to new situations. At present we consider gridworld-type problems where the goal remains constant across tasks but contingencies for reaching it can change.

II. ALGORITHM

A. *Hierarchical state abstraction to allow quick adaptation*

We use a hierarchical representation of space to facilitate efficient adaptation of previously learned models to new problems. The state hierarchy is created by merging adjacent gridworld states (Fig. 2) - a simple de facto state abstraction scheme used by Dayan and Hinton [13]. Each resulting level of abstraction represents the problem at a different degree of granularity: the lowest level considers world states directly and contains the most detail, while the highest level represents the world using a few very general macro-states.

A separate model-based RL algorithm runs at each level of the hierarchy, learning the task at its own level of resolution. Each experience the agent gains in the world is represented as a tuple: <*s*,*a*,*s'*,*r*>, which indicates that action *a* was taken from state *s*, resulting in new state *s'* and reward *r*. The RL algorithm at the lowest level of the hierarchy (the world level) updates its models of *P(s'|s,a)* and *R(s,a,s')* after every experience, but the higher levels are updated only when a state transition is made *at their level* - in the meantime, the reward obtained while the agent

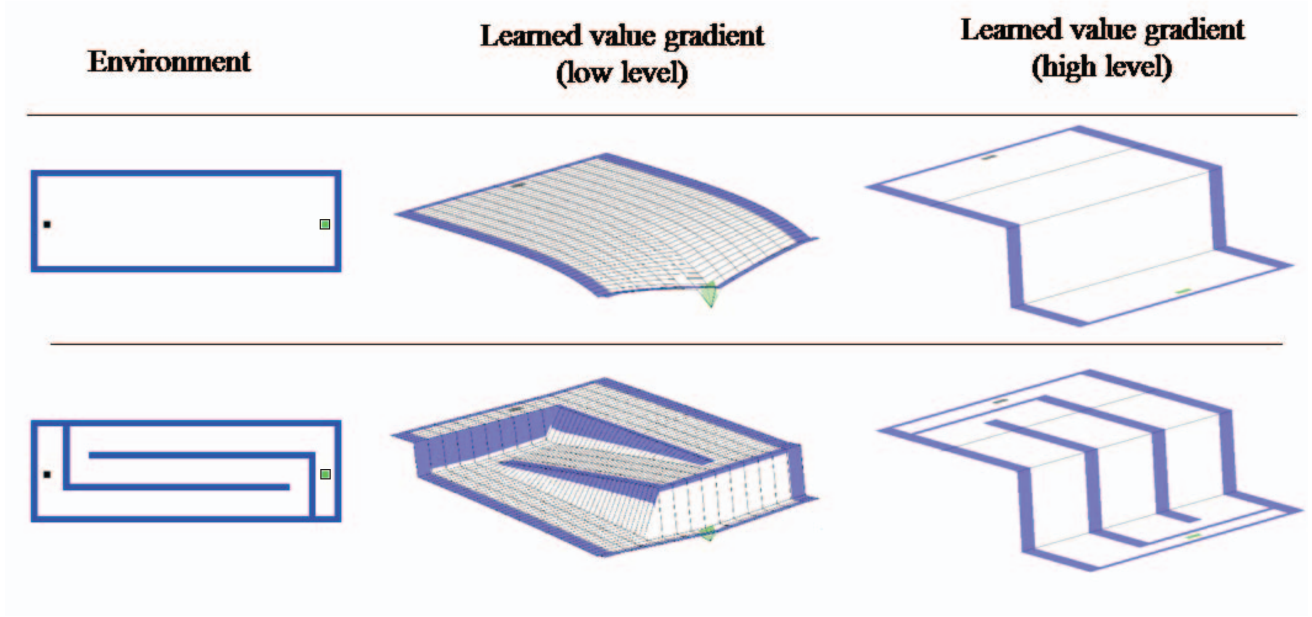
Environment
Learned value gradient (low level)
Learned value gradient (high level)

Fig. 1. Comparison of value gradients learned at low and high levels of abstraction. The agent learns to follow the gradient toward the goal. Environmental changes can drastically alter the gradient at the world-level, but at the highest level of abstraction the task rarely changes.

occupies a macro-state is aggregated. Thus models at each level generally require about half as many model updates as at the level below. Here we implement model-based RL using tabular models and Moore and Atkeson's prioritized sweeping algorithm [19].

This hierarchical structure facilitates an efficient planning process that can quickly adapt to a change in transition probabilities. A sudden and significant change to transition probabilities invalidates the value gradient at the lowest level, as illustrated in Fig. 1, but the changes have little effect on the value gradient at the highest level of abstraction, which is only aware of the general 'direction' of the goal (see Fig. 3). Thus, the highest level can use its action values to select a goal state (a macro state at its own level of abstraction), while the lower levels plan a route to this goal. Route finding does not rely on the (possibly invalid) action values, but rather on the model of *P(s'|s,a)*, which quickly incorporates new transition possibilities as the agent experiences them. Thus the burden of decision making (which requires accurate action values) is shifted to the higher levels of abstraction, while the lower detail-rich levels contribute world-level plans to achieve the high-level goals. The upshot of this is that the hierarchical system can adapt a previously-learned model to a new environment.

The hierarchical goal-setting and planning process (illustrated in Fig. 4) proceeds as follows. The level of abstraction with the highest available action value (let this level be level n) is allowed to set the goal - the goal state being the state (or macro-state) where the corresponding action is expected to lead. This goal state is mapped to four states at level n-1, and the model at this level plans a route from the current state to one of these four states, at its own level of abstraction. The first state in this route maps to four states at level n-2, and so on. This recursive planning process produces a hierarchical plan, with the steps of the plan existing at progressively higher levels of abstraction as the plan extends into the future. Since the number of states decreases exponentially with the level of abstraction, planning in this way is more efficient than creating an entire plan at the world level, especially if changing transition possibilities can invalidate the plan before its completion. Our current implementation uses the A* search algorithm for route finding, using action values as a heuristic in the search.

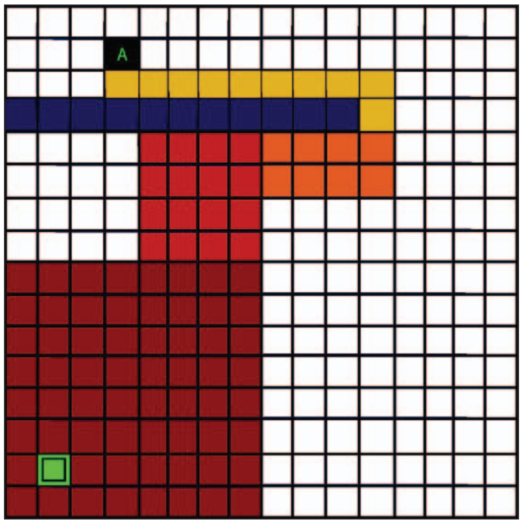

Fig. 4. Illustration of the hierarchical planning process, in which the agent (marked with a letter 'A') must reach the goal (marked in green) by navigating around a boundary (blue squares). A hierarchical plan is created for reaching the high-level goal state in the lower-left corner. Shades of red indicate the level of abstraction of various stages of the plan.

One important note is that the discount factor γ (in Eq. 2) must decrease as one moves up the abstraction hierarchy. This ensures that the highest available action value will appear at lower and lower levels as the agent approaches the goal, allowing the agent to zero-in on the goal rather than becoming trapped operating at a high level of abstraction.

B. *Context switching: detecting changes in the environment*

The agent maintains a history of its recent state transitions, and a library of saved models. It constantly checks its state transition history against its current model: if the model no longer explains it satisfactorily, the agent assumes the environment has changed and saves its current model for future use. It can then retrieve from the library a previously saved model that better predicts the recent state transitions.

The agent maintains a history **H** of its *m* most recent state transitions: $\mathbf{H} = \{t_0, t_1, \dots, t_{m-1}\}$, where each state transition $t_i = \langle s_i, a_i, s'_i, r_i \rangle$. Our implementation of the model for *R(s,a,s')* tracks the distribution of rewards rather than the average reward for a given transition. Thus, the *P(s'|s,a)* and *R(s,a,s')* models together can estimate the probability of a state transition as the joint probability: $P(t_i) = P(s'_i, r_i \mid s_i, a_i)$. The probability of **H** under the current models can be estimated:

$$P(\mathbf{H}) = \prod_{i=0}^{m-1} P(\mathrm{t}_i) \tag{3}$$

This assumes that the probabilities *P*(t) are independent of each other, (i.e. that the recent state transitions do not change the models significantly). One practical interpretation of this could be that the agent is allowed to learn a new task completely before being switched to a different task. We leave to future work the more challenging situation in which task changes can occur before the task is even learned.

At each step, the agent operates using whichever of its saved models gives the highest *P(***H***)*. A model change, if appropriate, is performed before using the current state transition for a model update. If all *P(***H***)* values fall below a set threshold (i.e. none of the saved models explains the recent state transitions satisfactorily) the agent assumes it has been presented with a new environment.

C. *Combining context switching with hierarchy-based adaptation for transfer learning*

We present a system which employs the context switching and hierarchical state abstraction mechanisms described above. Context switching allows the agent to retrieve previously learned model-hierarchies relevant to a new environment . If no saved model-hierarchies apply, the hierarchical architecture is used to adapt the closest match to the new environment. If the adaptation fails, a new model is created to represent the

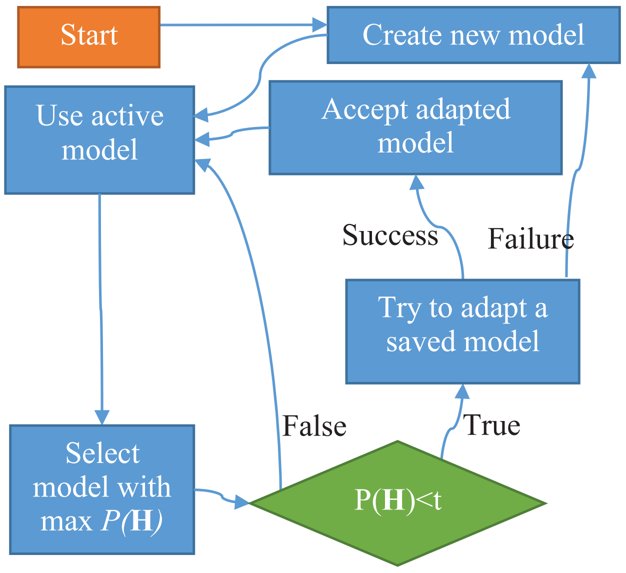

Fig. 5. State machine for the learning system combining hierarchy-based adaptation with context switching. When none of the previously learned models is suitable for a new environment, the agent attempts to adapt one. If adaptation fails it learns a new model to represent the environment.

environment. The general operation is illustrated in Fig. 5, and described in detail below.

The agent maintains a library of model-hierarchies. During operation one of these model-hierarchies is active and its lowest level model controls the agent's decision making. The higher levels only participate in decision making during the adaptation phase described below, but are continuously learning from the agent's experiences. After each step the agent calculates *P(***H***)* values for each model-hierarchy in the library. Note that the *P(***H***)* values are calculated using the *P(s'|s,a)* and *R(s,a,s')* models at the lowest level of each model-hierarchy. The model-hierarchy providing the highest *P(***H***)* value is set as active and is updated with the state transition tuple from the current step.

If all *P(***H***)* values fall below a threshold *t*, this indicates that none of the models can explain the recent state transition history. The agent first tries to adapt a model from its library to (what is assumed to be) the new task. It makes a temporary copy of the model-hierarchy with the best *P(***H***)* value, now invoking the full hierarchical planning process in attempt to find a solution. A new model-hierarchy is also created, which we will call the 'tentative' model-hierarchy. While the copied model-hierarchy controls the agent during the attempted adaptation, the tentative model simply observes and learns from the agent's experiences. This procedure prevents mixing of information relevant to the separate environments.

Adaption is considered successful if the agent reaches the goal state: the tentative model is accepted into the library and set as active, while the temporary copy (which now contains a mix of information relating to the original and new environments) is deleted. The adaptation is considered to have failed if the

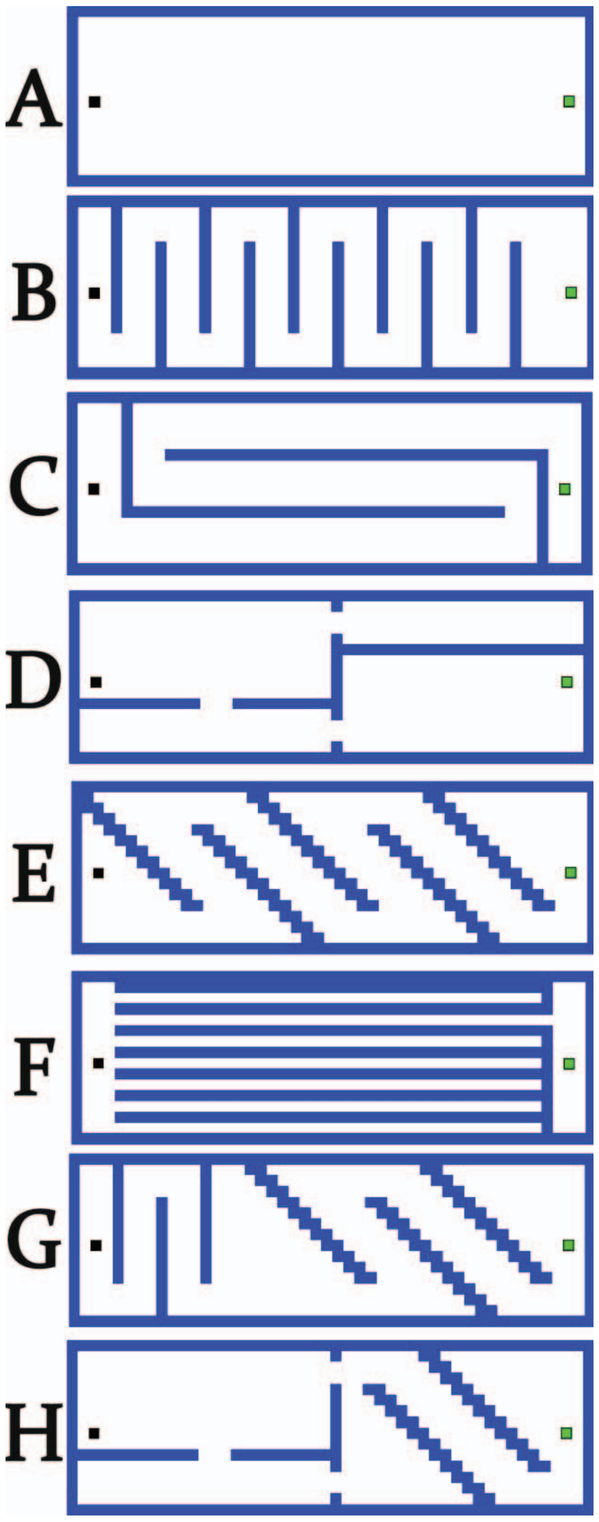

Fig. 6. Gridworld environments used for testing. The agent starts at the state marked in black, and must reach the goal state marked in green, while avoiding the walls shown in blue.

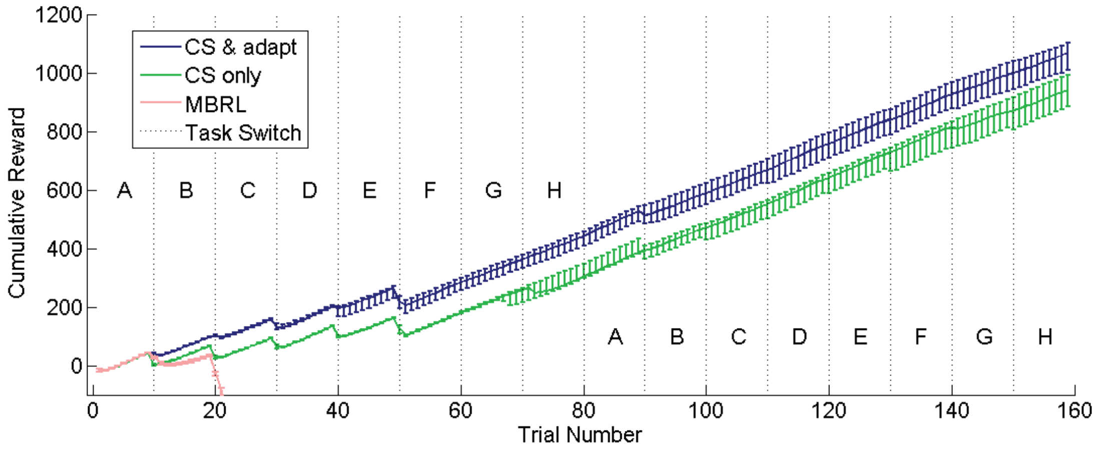

Fig. 2. Cumulative reward obtained during the changing-task test by three learning methods: context-switching with hierarchical adaptation (CS & adapt), context-switching only (CS only), and a standard model-based reinforcement learning algorithm (MBRL). Context-switching provides the ability to learn and recall solutions to multiple tasks, while hierarchical adaptation accelerates learning when the task is first encountered. Plots show median cumulative reward, with error bars showing 25th and 75th percentile values.

hierarchical planning process fails, or the value gradient from the agent's current state completely flattens. In this case the adaptation is aborted and the tentative model must learn the entire new environment.

III. Testing

Three RL approaches were compared: the system of context switching with hierarchical adaptation, context switching with no hierarchical adaptation, and conventional model-based RL. Each was tested using the series of gridworld environments shown in Fig. 6. After ten trials in each environment, the agent was switched to the next environment and forced to learn the new solution. Each "trial" began with the agent at the starting state, and ended when the agent reached the goal state or after 5000 steps. The sequence of eight environments was repeated twice. Each gridworld was 16 by 48 states. Navigating into a wall triggered a reward of -0.1, finding the goal triggered a reward of 10, and there was a small negative reward of -0.01 for all other transitions.

All agents used an ϵ-greedy policy, in which the action with the highest value was selected with probability ϵ, and a random action was chosen otherwise. The parameter ϵ was set to 0.9, the transition history length was set to 20, and the probability threshold t was set to 0.05. Each test was repeated 200 times. The conventional model-based algorithm was allowed up to 120 model updates per step, while the hierarchical system was allowed 20 updates per step per layer, for a total of 120 across its six layers. The discount factor γ (in Eq. 2) was set to 0.9 at the lowest level of abstraction, and to 0.4 at all higher levels. All parameters were empirically tuned for good performance in mazes A and B, and held constant during other testing; an analysis of the sensitivity of our approach to ϵ and γ selections would be problem-dependent and is not included in this work.

To further illustrate the adaptation and context switching functions, a second test was performed in which the agent was exposed to environments B, E, and G. The effect of exposing the agent to environment A prior to this sequence was examined.

IV. Results

Fig. 7 shows the cumulative reward obtained by the system of context switching with adaptation, context switching without adaptation, and a conventional model-based RL algorithm, over two repetitions of the eight sequential environments. Context switching allows an agent to learn solutions to each environment, and retrieve them at the second exposure to the environment. The hierarchy-based adaptation allows it to learn some solutions more quickly. Thus, combining context switching with adaptation achieved the best performance. The conventional model-based RL cannot cope with the environmental changes, due to the problem illustrated in Fig. 1.

Fig. 8 shows cumulative reward over the sequence of environments B, E, and G, with and without prior exposure to

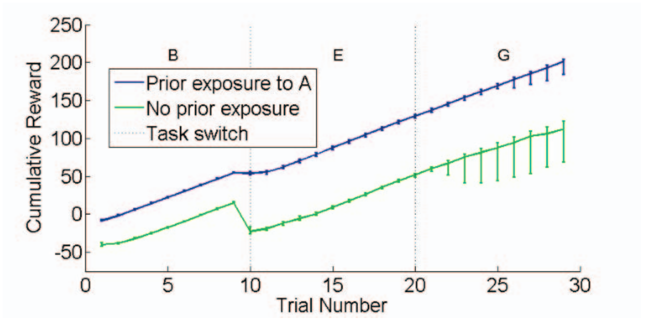

Fig. 8. Cumulative reward obtained by the system of context-switching with hierarchical adaptation, when exposed to environments B, E, and G in sequence. A model of environment B cannot be adapted to environment E, but a model of environment A can. Thus, prior exposure to environment A allows adaptation to environment E and reduces the drop in cumulative reward at trial number 10.

environment A. Environment A is compatible with adaptation to environments B and E. However environment B cannot be adapted to environment E, because of the problem illustrated in Fig. 1. Thus the agent with prior exposure to environment A achieves better performance, because the model for environment A can be adapted to solve environments B and E.

Fig. 9 shows the agent's trajectory in environment G after prior exposure to environments B and E. Since environment G can be solved through a combination of the models learned for environments B and E, the agent need not learn anything new.

V. Discussion

One of the most important aspects of intelligent behaviour is the capacity to learn from previous experience and adapt to changes in the environment. Even though there has been significant progress in the Artificial Intelligence field in this regard, the ability to "learn to learn" is still a major challenge. In this paper, a traditional RL system is expanded using a brain-inspired approach to solve a navigation task in a non-stationary environment. The proposed model uses a hierarchical structure inspired by the hippocampal formation. The hippocampus represents space using a hierarchical structure, in which the scale of the spatial location increases in direct proportion to the level of the structure where its representation is located. In addition, the hippocampus uses its 'spatial map' to represent the animal's location in the environment. If the environment changes slightly, the hippocampus modifies its current map of the world to account for the changes. However, if the animal is placed in a new environment, so that the changes are significant compared to the previous arena, the hippocampus uses a new 'map' for this new environment. This phenomenon is called global remapping [17]. Our approach takes advantage of both types of modifications.

In this work, a hierarchical spatial representation is used to solve a simulated navigation task . Progressively higher (more abstract) levels of the hierarchy view the locations in the arena at coarser spatial resolution, analogous to the hierarchical representation of space in the hippocampus. The hierarchical representation reduces search-space size at high levels of abstraction, allowing efficient hierarchical planning and adaptation to changes in the agent's environment. Context-switching is also implemented, allowing the agent to navigate to the goal in multiple environments. An environmental change

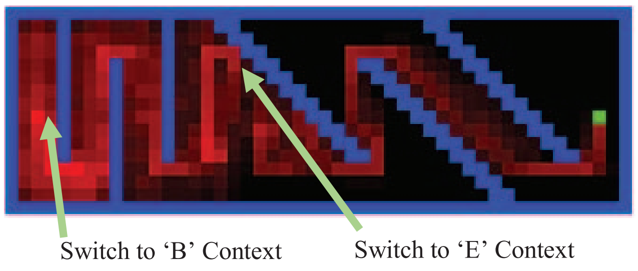

Fig. 9. An agent's trajectory in environment G, after prior exposure to environments B and E - Brighter red indicate states visited more often. The agent solves environment G by combining models learned in environments B and E.

prompts the agent to learn a new model representing the modified environment. Previously learned models are stored and can be retrieved in the future when needed. The effect of switching between learned models is analogous to the global remapping effect in the hippocampus.

The combination of context switching with hierarchical spatial representation provided a significant performance improvement over a traditional RL approach in this task. Fig. 8 illustrates that adapting a previously learned model can accelerate learning in a new environment. Fig. 9 shows that context switching can allow knowledge gained in one environment to be recycled, not only when the agent navigates this environment, but also when the knowledge is relevant to a new environment.

The current implementation of our proposed approach has several limitations. The issue of hierarchical optimality [25] arises whenever abstraction is used: Our hierarchical abstraction system allows quick adaptation, but learning to solve an abstraction of a problem can cause the learned behavior (which is optimal for the abstraction) to be suboptimal for the original problem. Moreover, the benefits of hierarchical adaptation are only realized when the agent's library contains a model which *can* be adapted to the new environment. Similarly, solving a new environment through context switching (as in Fig. 9) is only possible if the agent has the right 'building blocks' in its library. These features suggest that the agent's performance would continue to improve with exposure to varied environments, but if changes in the environment are large and frequent so that new models are required very often, or in environments with a large number of states, the expense of storing many models can be problematic.

The advantages of adaptation and context switching depend not only on the agent having appropriate models in its library, but also on correct identification of the right model at the right time. Random actions during the first few steps in a new environment can affect the agent's prediction of which MDP it is in. If the wrong model is selected, the agent will end up learning a new model – potentially duplicating information in its library. Theoretically, the sequence of environments shown in Fig. 6 can be solved using only six models (modelling environments A through F, with environments G and H being solvable using combinations of B, D, and E. Error bars in Fig. 7 show a range of performance: agents who performed best learned the minimum number of models, while the poorer performers learned more (16 separate models in the worst case). There is room to improve the mechanism by which the agent selects a model from its library. Alternatively, performance might be improved by incorporating Lazaric's method of transferring single state transitions rather than an entire model [21]. It is thought though, that the brain maintains multiple models simultaneously, switching seamlessly between them when the information they contain becomes relevant and it receives the appropriate feedback [26], [27].

This work has considered transfer learning across problems with identical goal states, and similar state and action spaces. The hierarchical planning process is also currently restricted to mostly deterministic environments. Future work should consider transfer learning in MDP problems more generally. This consideration will involve a method of automatic state-space abstraction (as opposed to the de facto scheme illustrated in Fig. 2).

VI. Conclusion

Complementing traditional RL approaches with a brain-inspired hierarchical structure and context-switching mechanism shows significant advantages in the face of environmental changes. In this work the strategy was applied to a navigation task, but the same approach could be transferable to other domains where the state space of the task could be dissected in a similar manner. Finding adaptive solutions to problems in dynamic environments is crucial in the pursuit of transfer learning, and in the design of intelligent agents.

Acknowledgment

The authors would like to thank Adam Neumann for helpful discussion.

References

[1] S. Thrun and L. Pratt, Learning to Learn. Springer Science & Business Media, 2012.

[2] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. MIT Press, 1998.

[3] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in Proceedings of the seventh international conference on machine learning, 1990, pp. 216–224.

[4] K. Doya, "Reinforcement learning: Computational theory and biological mechanisms," HFSP J., vol. 1, no. 1, pp. 30–40, May 2007.

[5] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," J. Mach. Learn. Res., vol. 10, pp. 1633–1685, 2009.

[6] D. Precup, "Temporal abstraction in reinforcement learning," Ph.D., University of Massachusetts, Amherst, MA, 2000.

[7] M. Stolle and D. Precup, "Learning options in reinforcement learning," in Lecture Notes in Computer Science, 2002, pp. 212–223.

[8] M. M. Botvinick, "Hierarchical reinforcement learning and decision making," Curr. Opin. Neurobiol., vol. 22, no. 6, pp. 956–962, Dec. 2012.

[9] M. M. Botvinick, Y. Niv, and A. C. Barto, "Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective," Cognition, vol. 113, no. 3, pp. 262–280, Dec. 2009.

[10] T. G. Dietterich, "The MAXQ Method for Hierarchical Reinforcement Learning.," in Proceedings of the 15th International Conference on Machine Learning, San Mateo, CA, 1998, pp. 118–126.

[11] J. Morimoto and K. Doya, "Hierarchical Reinforcement Learning of Low-Dimensional Subgoals and High-Dimensional Trajectories.," in Proceedings of the 5th International Conference on Neural Information Processing, Burke, VA, 1998, pp. 850–853.

[12] C. Diuk, A. L. Strehl, and M. L. Littman, "A hierarchical approach to efficient reinforcement learning in deterministic domains," in Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, 2006, pp. 313–319.

[13] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in Advances in neural information processing systems, 1993, pp. 271–271.

[14] T. Mao and L. Ray, "Hierarchical State Representation and Action Abstractions in Q-Learning for Agent-Based Herding," Int. J. Inf. Electron. Eng., vol. 2, pp. 538–542, 2012.

[15] M. W. Jung, S. I. Wiener, and B. L. McNaughton, "Comparison of spatial firing characteristics of units in dorsal and ventral hippocampus of the rat," J. Neurosci., vol. 14, no. 12, pp. 7347–7356, Dec. 1994.

[16] A. Botea, M. Müller, and J. Schaeffer, "Near optimal hierarchical path-finding," J. Game Dev., vol. 1, no. 1, pp. 7–28, 2004.

[17] M.-B. Moser, D. C. Rowland, and E. I. Moser, "Place Cells, Grid Cells, and Memory," Cold Spring Harb. Perspect. Biol., vol. 7, no. 2, p. a021808, Feb. 2015.

[18] F. Chersi and N. Burgess, "The Cognitive Architecture of Spatial Navigation: Hippocampal and Striatal Contributions," Neuron, vol. 88, no. 1, pp. 64–77, Jul. 2015.

[19] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," Mach. Learn., vol. 13, no. 1, pp. 103–130, Oct. 1993.

[20] A. D. Redish, S. Jensen, A. Johnson, and Z. Kurth-Nelson, "Reconciling reinforcement learning models with behavioral extinction and renewal: implications for addiction, relapse, and problem gambling.," Psychol. Rev., vol. 114, no. 3, p. 784, 2007.

[21] A. Lazaric, "Transfer in reinforcement learning: a framework and a survey," in Reinforcement Learning - State of the art, Springer, 2012, pp. 143–173.

[22] N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern, "Transfer in variable-reward hierarchical reinforcement learning," Mach. Learn., vol. 73, no. 3, pp. 289–312, 2008.

[23] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in In International Conference on Robotics and Automation, 1997.

[24] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, 2006, pp. 720–727.

[25] B. Hengst, "Hierarchical Reinforcement Learning," in Encyclopedia of Machine Learning, C. Sammut and G. I. Webb, Eds. Springer US, 2011, pp. 495–502.

[26] G. Dragoi and S. Tonegawa, "Preplay of future place cell sequences by hippocampal cellular assemblies," Nature, vol. 469, no. 7330, pp. 397–401, Jan. 2011.

[27] J. P. Gallivan, L. Logan, D. M. Wolpert, and J. R. Flanagan, "Parallel specification of competing sensorimotor control policies for alternative action options," Nat. Neurosci., vol. 19, no. 2, pp. 320–326, Feb. 2016.