

# FALCON: A Fusion Architecture for Learning, COgnition, and Navigation

Ah-Hwee Tan

School of Computer Engineering  
Nanyang Technological University  
Nanyang Avenue, Singapore 639798  
E-mail: asahtan@ntu.edu.sg

**Abstract**—This paper presents a natural extension of self-organizing neural network architecture for learning cognitive codes across multi-modal patterns involving sensory input, actions, and rewards. The proposed cognitive model, called FALCON, enables an autonomous agent to adapt and function in a dynamic environment. Simulations based on a minefield navigation task indicate that the system is able to adapt amazingly well and learns rapidly through its interaction with the environment in an online and incremental manner. The scalability and robustness of the system is further enhanced by an online code evaluation and pruning procedure, that maintains the number of cognitive codes at a manageable size without degradation of system performance.

## I. INTRODUCTION

Autonomous agents in general refer to organisms that are able to function and adapt by themselves in a complex and dynamic environment. Key characteristics of autonomous agents including reasoning, continuous real-time learning, ability to reflect and explain, self-awareness, and handling of surprises. In modern cognitive science, many have held the view that cognition is a process deeply rooted in the body's interaction with the world [1]. In other words, autonomous systems acquire intelligence through their interaction with the environment. In addition, very simple computing mechanism, such as competition and clustering, can give rise to high level complex behaviour [2].

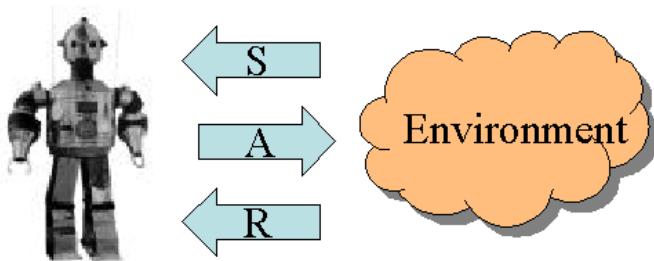


Fig. 1. The embodied agent paradigm. An autonomous system receives sensory input (S) from its environment, selects and performs an action (A), and receives feedback (R) from the environment.

As described in the field of reinforcement learning [3], an autonomous agent typically operates in a sense, act, and learn cycle (Figure 1). An autonomous system obtains sensory input from its environment in the form of current state (S).

Depending on the current state and its knowledge and drive, the system selects and performs the most appropriate action (A). Upon receiving the feedback in terms of rewards (R) from the environment, the system learns to adjust its behaviour in the motivation of receiving positive rewards in the future.

Classical solutions to the reinforcement learning problem involve learning at least two mappings, the first mapping a state and an action to a new state, and the second mapping a state to a utility value, using strategies in reinforcement learning, such as value iteration and Q-learning [4]. The problem of this approach is that mappings must be learned for each and every possible pair of states and actions. This causes a scalability issue for a very large state space. Although there have been attempts in using machine learning methods, such as backpropagation neural networks [5], [6] and decision tree algorithms [7] to learn the compressed mappings, these learning methods are typically slow and require a training phase. As a result, these systems will not be able to learn and operate in real time.

Self-organizing neural networks, such as Adaptive Resonance Theory (ART) models [8], are capable of learning recognition categories of multi-dimensional mappings of input patterns in an online and incremental manner. They are thus suitable candidates for building autonomous systems. However, whereas various models of ART and their supervised learning versions [9], [10] have been widely applied to pattern analysis and recognition tasks, there have been very few attempts to use ART-based networks for building autonomous systems.

This paper describes an extension of ART for developing a cognitive model that learns and adapts its behaviour through the interaction with the environment. Whereas predictive ART performs supervised learning through the pairing of teaching signals and the input patterns, the proposed neural architecture, known as FALCON (Fusion Architecture for Learning, COgnition, and Navigation), learns multiple mappings simultaneously across multi-modal input patterns, involving states, actions, and rewards, in an online and incremental manner. Specifically, the system learns to adapt its behaviour through the learning of cognitive codes associating states to actions, that lead to desirable outcomes (rewards). Whereas a positive feedback reinforces the selected action, negative experience results in a reset, following which the system seeks

alternate actions. The idea is to learn to associate a state with an action that will lead to a desirable outcome. Using competition and clustering as the basis of computation, the network dynamics encompasses a myriad of learning paradigms, including unsupervised learning, supervised learning, as well as reinforcement learning.

To evaluate the proposed system, experiments are conducted based on a minefield navigation task similar to one developed at the Naval Research Laboratory (NRL) [7]. The task involves controlling an autonomous vehicle (AV) that has to avoid mines and learning to navigate through obstacles and reach a stationary target (goal) within a specified number of steps. Experimental results show that using the proposed cognitive model, the AV adapts amazingly well and learns to perform the task rapidly through experience in an online manner. To enable the system to function in a complex and dynamic environment, an adaptive pruning algorithm is incorporated to evaluate and remove excessive cognition codes. The pruning algorithm has been effective in capping the number of cognitive codes within a reasonable size with no degradation of system performance.

The rest of the paper is organized as follows. Section 2 presents the cognitive model and the associated algorithms for action selection and learning. Section 3 describes the minefield navigation simulation task. Section 4 outlines the experiment methodology and discusses the simulation results. Section 5 analyzes the complexity of the problem domain and presents the adaptive code pruning algorithm. Section 6 summarizes and provides a discussion of the related and future work.

## II. ARCHITECTURE

The proposed cognitive model known as FALCON is based on multi-channel Adaptive Resonance Associative Map (multi-channel ARAM) [11], an extension of predictive Adaptive Resonance Theory (ART) networks. Whereas predictive neural network models, such as ARTMAP network [9] and ARAM [10], learn multi-dimensional mappings between the input and output patterns, multi-channel ARAM formulates cognitive codes associating multi-modal patterns across multiple input channels.

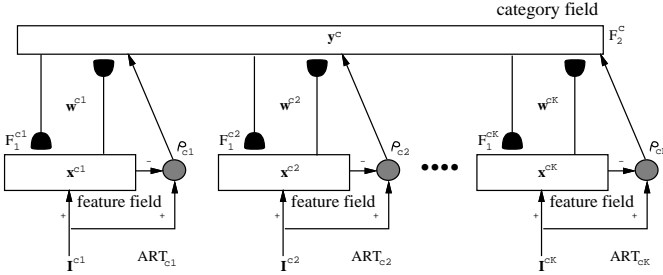


Fig. 2. The architecture of multi-channel ARAM.

A multi-channel ARAM model consists of a number of input representation fields  $F_1^{c1}, F_1^{c2}, \dots, F_1^{cK}$  and a category field  $F_2^c$  (Figure 2). 2-channel ARAM is equivalent to ARAM. If only one channel is present, multi-channel ARAM reduces

to ART. The dynamics of FALCON based on 3-channel ARAM, that employs fuzzy ART operations [12], is described below.

**Input vectors:** Let  $\mathbf{S} = (s_1, s_2, \dots, s_n)$  denote the state vector, where  $s_i$  indicates the sensory input  $i$ . Let  $\mathbf{A} = (a_1, a_2, \dots, a_m)$  denote the action vector, where  $a_i$  indicates a possible action  $i$ . Let  $\mathbf{R} = (r, \bar{r})$  denote the reward vector, where  $r \in [0, 1]$  and  $\bar{r} = 1 - r$ .

**Activity vectors:** Let  $\mathbf{x}^{ck}$  denote the  $F_1^{ck}$  activity vector. Let  $\mathbf{y}^c$  denote the  $F_2^c$  activity vector.

**Weight vectors:** Let  $\mathbf{w}_j^{ck}$  denote the weight vector associated with the  $j$ th node in  $F_2^c$  for learning the input representation in  $F_1^{ck}$ . Initially, all  $F_2^c$  nodes are uncommitted and the weight vectors contain all 1's.

**Parameters:** The FALCON's dynamics is determined by choice parameters  $\alpha^{ck} > 0$  for  $k = 1, \dots, K$ ; learning rate parameters  $\beta^{ck} \in [0, 1]$  for  $k = 1, \dots, K$ ; contribution parameters  $\gamma^{ck} \in [0, 1]$  for  $k = 1, \dots, K$  where  $\sum_{k=1}^K \gamma^{ck} = 1$ ; and vigilance parameters  $\rho^{ck} \in [0, 1]$  for  $k = 1, \dots, K$ .

### A. From Sensory to Action

Given the state vector  $\mathbf{S}$ , the system performs code competition and selects an action based on the output activities of action vector  $\mathbf{A}$  (Figure 3). The detailed algorithm is presented below.

Upon input presentation,  $\mathbf{x}^{c1} = \mathbf{S}$ ,  $\mathbf{x}^{c2} = \mathbf{N}$ , where  $N_i = 1$  for all  $i$ , and  $\mathbf{x}^{c3} = (1, 0)$ .

**Code activation:** Given activity vectors  $\mathbf{x}^{c1}, \mathbf{x}^{c2}, \dots, \mathbf{x}^{cK}$ , for each  $F_2^c$  node  $j$ , the choice function  $T_j^c$  is computed as follows:

$$T_j^c = \sum_{k=1}^K \gamma^{ck} \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_j^{ck}|}{\alpha^{ck} + |\mathbf{w}_j^{ck}|}, \quad (1)$$

where the fuzzy AND operation  $\wedge$  is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i), \quad (2)$$

and the norm  $|\cdot|$  is defined by

$$|\mathbf{p}| \equiv \sum_i p_i \quad (3)$$

for vectors  $\mathbf{p}$  and  $\mathbf{q}$ .

**Code competition:** All  $F_2^c$  nodes undergo a code competition process. The winner is indexed at  $J$  where

$$T_J^c = \max\{T_j^c : \text{for all } F_2^c \text{ node } j\}. \quad (4)$$

When a category choice is made at node  $J$ ,  $y_J^c = 1$ ; and  $y_j^c = 0$  for all  $j \neq J$ .

**Action selection:** The chosen  $F_2^c$  node  $J$  performs a readout of its weight vector to the action field  $F_1^{c2}$  such that

$$\mathbf{x}^{c2} = \mathbf{w}_J^{c2}. \quad (5)$$

The chosen action  $a_I$  is then determined by

$$x_I^{c2} = \max\{x_i^{c2} : \text{for all } F_1^{c2} \text{ node } i\}. \quad (6)$$

### B. From Feedback to Learning

Upon receiving a feedback from its environment after performing the action  $a_I$ , the system adjusts its internal representation based on the following principles. Given a reward (positive feedback), the agent learns that an action executed in a state will result in a favorable outcome. Therefore, the system learns to associate the state vector  $\mathbf{S}$ , the action vector  $\mathbf{A}$ , and the reward vector  $\mathbf{R}$ . Conversely, if a penalty is received, there is a reset of action and the agent learns the mapping among the state vector  $\mathbf{S}$ , the complement of action vector  $\bar{\mathbf{A}}$  where  $\bar{a}_i = 1 - a_i$  for all  $i$ , and the complement of reward vector  $\bar{\mathbf{R}}$  where  $\bar{r}_i = 1 - r_i$  for all  $i$ .

Given the activity vectors  $\mathbf{x}^{c1}$ ,  $\mathbf{x}^{c3}$ , and  $\mathbf{x}^{c3}$ , the system performs code activation and code competition (as described in the previous section) to select a winner  $J$  for encoding the mapping.

**Template matching:** Before code  $J$  can be used for learning, a template matching process checks that the weight templates of code  $J$  are sufficiently close to their respective input patterns. Specifically, resonance occurs if for each channel  $k$ , the *match function*  $m_J^{ck}$  of the chosen code  $J$  meets its vigilance criterion:

$$m_J^{ck} = \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck}|}{|\mathbf{x}^{ck}|} \geq \rho^{ck}. \quad (7)$$

Learning then ensues, as defined below. If any of the vigilance constraints is violated, mismatch reset occurs in which the value of the choice function  $T_J^c$  is set to 0 for the duration of the input presentation. The search process repeats to select another  $F_2^c$  node  $J$  until resonance is achieved.

**Template learning:** Once a node  $J$  is selected for firing, for each channel  $k$ , the weight vector  $\mathbf{w}_J^{ck}$  is modified by the following learning rule:

$$\mathbf{w}_J^{ck(\text{new})} = (1 - \beta^{ck})\mathbf{w}_J^{ck(\text{old})} + \beta^{ck}(\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck(\text{old})}). \quad (8)$$

### III. MINEFIELD NAVIGATION SIMULATION

A simulator has been developed in-house for a minefield navigation task (Figure 4). The simulator allows a user to specify the size of the minefield as well as the number of mines in the field. In each trial, the autonomous vehicle (AV) starts at a randomly chosen position in the field. The objective is to navigate through the minefield to a randomly selected target position in a specified time frame without hitting a mine. The target and the mines remain stationary during the trial. A trial ends when the system reaches the target (success), hits a mine (failure), or runs out of time.

Minefield navigation and mine avoidance is a non-trivial task. As the configuration of the minefield is generated randomly and changes over trials, the system needs to learn strategies that can be carried over across experiments. In addition, the system has a rather coarse sensory capability with a 180 degree forward view based on five sonar sensors. The sonar signal in the  $i$  direction is measured by

$$s_i = \frac{1}{1 + d_i} \quad (9)$$

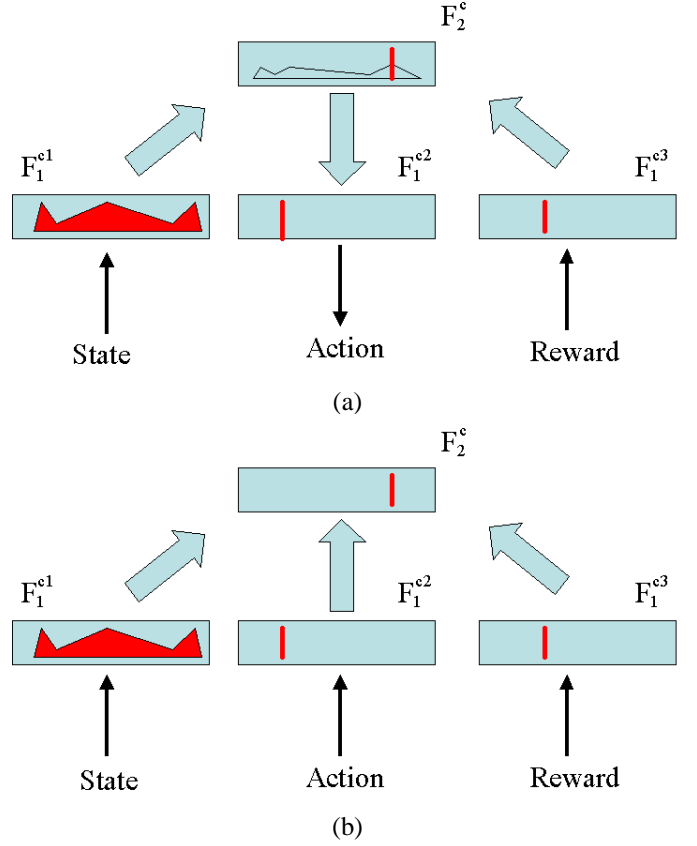


Fig. 3. The sense, act, and learn cycle of FALCON. (a) Given the state vector  $\mathbf{S}$ , the system performs code competition and produces an action vector  $\mathbf{A}$ . (b) Upon receiving a positive feedback, the system learns to associate the state vector  $\mathbf{S}$ , the action vector  $\mathbf{A}$ , and the reward vector  $\mathbf{R}$ .

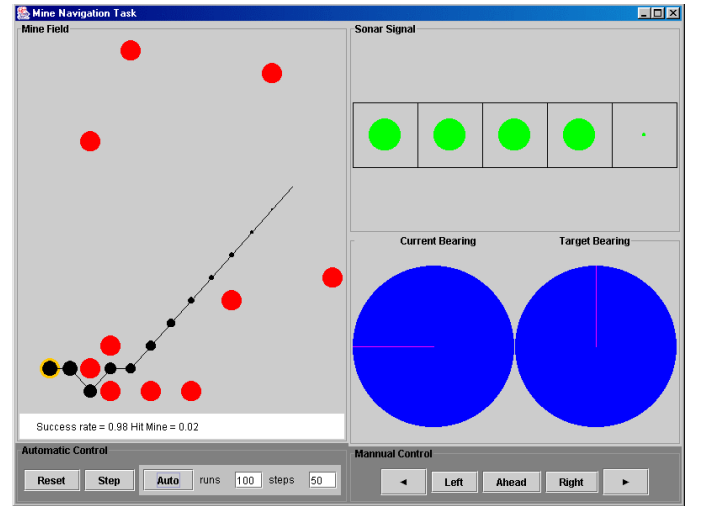


Fig. 4. The minefield navigation simulator.

where  $d_i$  is the distance to an obstacle (that can be a mine or the boundary of the minefield) in the  $i$  direction. Other input attributes of the sensory (state) vector include the range and the bearing of the target from the current position.

At each step, the system can choose one out of the five possible actions, namely move left, move diagonally left, move straight ahead, move diagonally right, and move right.

When a trial ends, a reward of 1 is given when the system reaches the target, whereas a reward of value 0 (punishment) is given when it hits a mine. During the trial, intermediate rewards are estimated by computing a utility function

$$utility = \frac{1}{1 + rd} \quad (10)$$

where  $rd$  is the remaining distance between the current position and the target position. A positive feedback is provided when the system improves the utility function over trials. Conversely, a negative feedback is given when there is no improvement in the utility value.

#### IV. EXPERIMENTAL RESULTS

Experiments were conducted based on a 16 by 16 minefield containing 10 mines. In each trial, the FALCON-based Autonomous Vehicle (AV) repeated the cycles of sense, move, and learn, until it reached the target, hit a mine, or exceeded 30 sense-move-learn cycles. Throughout the experiments, FALCON used the following set of parameter values: choice parameters  $\alpha^{ck} = 0.001$  and learning rate parameters  $\beta^{ck} = 1.0$  for  $k = 1, 2, 3$ ; contribution parameters  $\gamma^{c1} = 1.0, \gamma^{c2} = \gamma^{c13} = 0.0$ ; and vigilance parameters  $\rho^{c1} = 0.5, \rho^{c2} = 0.2, \rho^{c3} = 0.5$ .

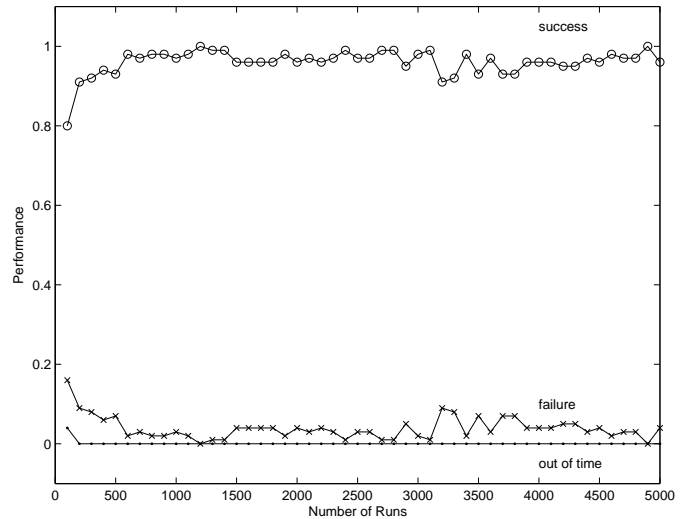
Figure 5(a) depicts the performance of the AV in terms of success rates, failure rates, and out-of-time rates. The success rates increase rapidly right from the start. By the end of 1000 trials, the AV has achieved close to 100% success rates.

To evaluate how well the AV traverses from its starting position to the target, we define a measure called *normalized step* given by

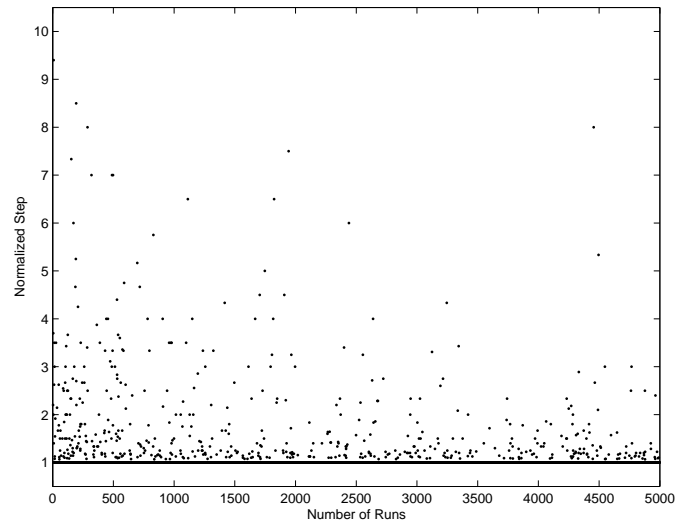
$$step_n = \frac{step}{sd} \quad (11)$$

where  $step$  is the number of sense-move-learn cycles taken to reach the target and  $sd$  is the shortest distance between the starting and target positions. A normalized step of 1 means the system has taken the optimal(shortest) path to the target. As depicted in Figure 5(b), after 1000 trials, the system has been able to reach the target in optimal or close to optimal paths in a great majority of the cases.

Figure 6 shows the exemplary paths taken by the system at runs 1, 100, 500, and 1000. At the beginning, the system clearly has no idea of how to reach the target and the path is basically random. By the 100<sup>th</sup> run, the system is able to get to the target somehow, but the path is still less than satisfactory. The system gradually improves its performance over time (as shown in trial 500). By the end of 1000 trials, optimal paths are always taken.



(a)



(b)

Fig. 5. Performance of the AV against the number of trials in terms of (a) success rate, failure rate, and out-of-time rate; and (b) normalized steps taken to the target.

#### V. ANALYSIS AND SCALABILITY

The complexity for learning the minefield navigation problem is largely determined by the dimension of the sensory(state) representation. The state space is  $S^5 \times B$  where  $S = [0, 1]$  is the range of the sonar signals and  $B = \{0, 1, \dots, 7\}$  is the set of possible target bearings. In the continuous form, this poses a great challenge for traditional reinforcement approaches. Even in binary case in which the sonar signals are binarized, there are  $2^5 * 8$  (more than 16 thousands) possible combinations.

As shown in Figure 7, using continuous sonar signal values, FALCON continues to create new cognitive codes over time. At the end of 5000 trials, over 2500 cognitive codes are created. This causes the system to slow down significantly. To handle this problem, we propose a method for evaluating

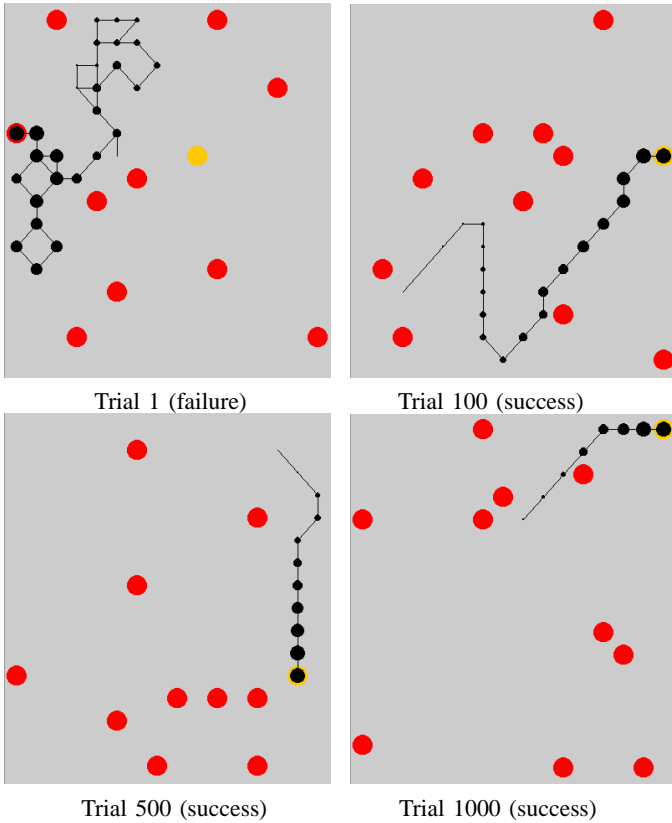


Fig. 6. Exemplary navigation paths of the AV at trials 1, 100, 500, and 1000.

and eliminating cognitive codes created by spurious cases during learning. Carpenter and Tan [13] used a similar confidence estimation scheme for pruning category nodes in fuzzy ARTMAP. The confidence values of the category nodes were computed based on their usage on a training set and their accuracy on a predicting set. The pruning was done in a batch mode *after learning* to reduce the rule set complexity for the purpose of interpretation. In the present problem, we aim to moderate the number of cognitive codes *during learning*. It follows that confidence estimation and code pruning have to be done in real time as part of the training process. Such a procedure is described below.

Each cognitive code  $j$  is assigned a confidence factor  $c_j$ , a real number between 0 to 1. For a newly learned code  $j$ ,  $c_j$  equals 1. At a fixed interval, a *forgetting* process causes  $c_j$  to decay towards zero. A reinforcing process increases  $c_j$  towards 1 whenever a correct prediction is made by the category  $j$  during learning. Similarly, a penalty process reduces  $c_j$  towards 0 whenever a negative feedback is received.

**Confidence decay:** At a constant interval, the confidence value of each cognitive code depreciates towards zero according to the equation

$$c_j^{(\text{new})} = c_j^{(\text{old})} - \zeta c_j^{(\text{old})}. \quad (12)$$

The decay is self-scaling in the sense that the decay becomes smaller as  $c_j$  gets smaller.

**Confidence reinforcement:** When the action specified by a

chosen cognition code  $J$  receives a positive feedback, its confidence value  $c_J$  is increased towards one by

$$c_J^{(\text{new})} = c_J^{(\text{old})} + \eta(1 - c_J^{(\text{old})}). \quad (13)$$

**Confidence erosion:** When the action of a chosen cognition code  $J$  receives a negative feedback, its confidence value  $c_J$  is decreased towards zero by

$$c_J^{(\text{new})} = c_J^{(\text{old})} - \theta c_J^{(\text{old})}. \quad (14)$$

**Category pruning:** The computed confidence values are then compared with a threshold parameter  $\tau$ . A code is removed when its confidence value falls below  $\tau$ . To prevent excessive pruning, removal of codes is only done when the number of cognitive codes exceeds a specified ceiling  $c$ .

Experiments were conducted using the following parameter values:  $\zeta = 0.0005$ ,  $\eta = 0.5$ ,  $\theta = 0.2$ , and  $\tau = 0.01$ . Figure 7 shows the number of cognitive codes created by the AV without adaptive pruning and with adaptive pruning at ceilings of 500 and 1000. Without adaptive pruning (top line), the number of cognitive codes keeps increasing as the number of trials increases. With adaptive pruning, the numbers of cognitive codes converge to the respective ceilings specified. More importantly, as shown in Figure 8, there is no noticeable degradation of performance although the number of cognition nodes has been greatly reduced through adaptive online pruning. Experiments were also conducted with binarized sonar signals with values of *near* and *far*, that greatly reduced the size of the state space. Similar performance was again obtained but with a much smaller number of less than 300 cognitive codes after 5000 trials.

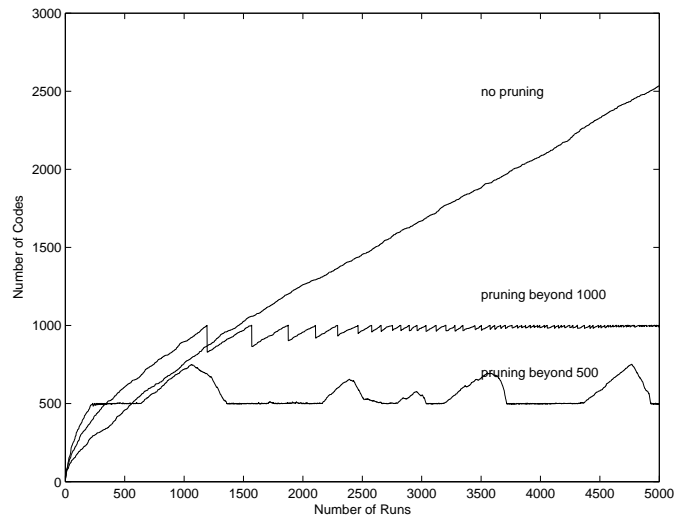


Fig. 7. Number of cognitive codes created against the number of trials without adaptive pruning and with adaptive pruning at ceilings of 500 and 1000.

## VI. DISCUSSION

We have presented a fusion architecture, known as FALCON, for learning multi-modal mappings across states, actions, and rewards. The proposed model provides a basic

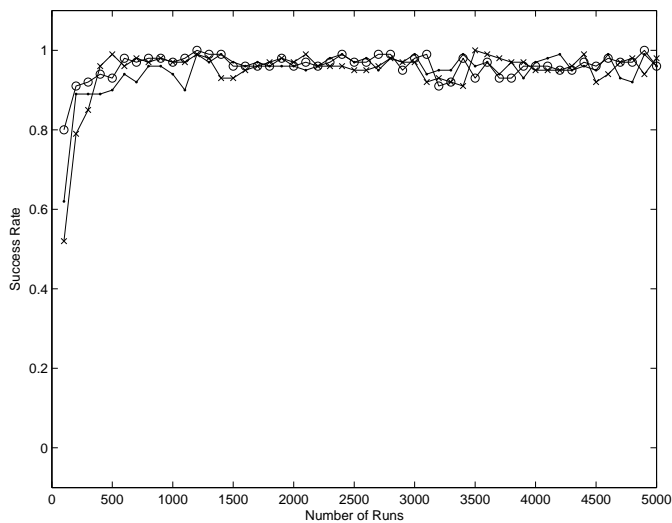


Fig. 8. Success rates against the number of trials without adaptive pruning and with adaptive pruning at ceilings of 500 and 1000.

building block for developing autonomous agents that are capable of functioning and adapting in a dynamic environment.

Learning of cognitive codes in FALCON is similar to the chunking process in SOAR architecture [14]. The fusion architecture of FALCON is consistent with those associated with mirror neurons [15], [16], a group of neurons which will fire when a subject is doing an action or observes someone else doing the same action. Mirror neurons are believed to serve as a mechanism for multi-modal (motor, perceptual and proprioceptive) information coordination and completion. As a natural extension of Adaptive Resonance Theory (ART), the underlying computation mechanism of FALCON is also supported by a rich literature of neurophysiological data [17].

The minefield simulation task described in this paper is similar to the NRL navigation and mine avoidance domain. To tackle the NRL problem, Gordan and Subramanian [7] developed two sets of cognitive models, one for predicting the next sonar and bearing configuration based on the current sonar and bearing configuration and the chosen action; and the other for estimating the desirability of the sonar and bearing configurations. Sun et. al. [18] used a three-layer Backpropagation neural network to learn the Q-values and an additional layer to perform stochastic decision making based on the Q-values. Compared with prior efforts, FALCON differs by the fact that associations are learned across states, actions, and rewards simultaneously using an integrated fusion model. More importantly, whereas existing navigation models typically involve a training phase and a testing phase, there is no such distinction in FALCON.

Our experiments estimated intermediate rewards based on the remaining distances between the AV and the targets. This strategy, as recommended by Kaelbling et. al. [4], has greatly improved the system performance. For handling problems in which intermediate rewards are not available, the model could be extended to estimate delayed rewards using reinforcement

learning strategies such as Q-learning or temporal difference method. This will be a subject of our future investigation.

#### ACKNOWLEDGMENT

The research is supported in part by start up grant CE-SUG 07/03 from the School of Computer Engineering, Nanyang Technological University.

#### REFERENCES

- [1] M. A. Arbib, *The Handbook of Brain Theory and Neural Networks*, MIT Press, 2002.
- [2] L. A. Coward, "The recommendation functional architecture as the basis for a neurophysiological understanding of cognition," in *Understanding Representation*, 1995, pp. 1–13.
- [3] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA: MIT Press, 1998.
- [4] L.P. Kaelbling, M.L. Littman, and A.W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
- [5] D.H. Ackley and M.L. Littman, "Generalization and scaling in reinforcement learning," in *Advances in Neural Information Processing Systems 2*, 1990, pp. 550–557, San Mateo, CA: Morgan Kaufmann.
- [6] R.S. Sutton, *Temporal Credit Assignment in Reinforcement Learning*, Ph.D. Thesis, University of Massachusetts, Amherst, MA, 1984.
- [7] D. Gordan and D. Subramanian, "A cognitive model of learning to navigate," in *Nineteenth Annual Conference of the Cognitive Science Society*, 1997.
- [8] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54–115, June 1987.
- [9] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698–713, 1992.
- [10] A. H. Tan, "Adaptive Resonance Associative Map," *Neural Networks*, vol. 8, no. 3, pp. 437–446, 1995.
- [11] A. H. Tan and H. S. Soon, "Concept hierarchy memory model: A neural architecture for conceptual knowledge representation, learning, and commonsense reasoning," *International Journal of Neural Systems*, vol. 4, no. 2, pp. 93–124, 1996.
- [12] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.
- [13] G. A. Carpenter and A. H. Tan, "Rule extraction: From neural architecture to symbolic representation," *Connection Science*, vol. 7, no. 1, pp. 3–27, 1995.
- [14] P. Rosenbloom, J. Laird, and A. Newell, *The SOAR papers: research on integrated intelligence*, Cambridge, MA: MIT Press, 1993.
- [15] M. Jeannerod and J. Decety, "Mental motor imagery: a window into the representational stages of action," *Current Opinion in Neurobiology*, vol. 5, pp. 727–732, 1995.
- [16] J. Wiedermann, "Mirror neurons, embodied cognitive agents and imitation learning," in *Proceedings, ECML/PKDD*, 2003.
- [17] R. Raizada and S. Grossberg, "Towards a theory of the laminar architecture of cerebral cortex: Computational clues from the visual system," *Cerebral Cortex*, vol. 13, pp. 200–213, 2003.
- [18] R. Sun, E. Merrill, and T. Peterson, "From implicit skills to explicit knowledge: a bottom-up model of skill learning," *Cognitive Science*, vol. 25, no. 2, pp. 203–244, 2001.