CS 885 Reinforcement Learning

Neural Map: Structured Memory for Deep Reinforcement Learning

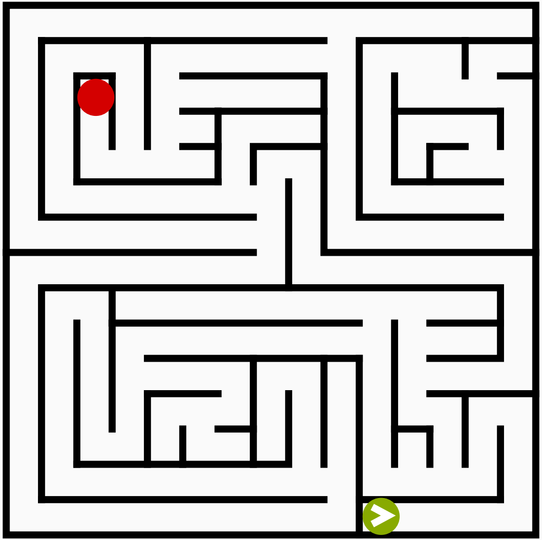
Emilio Parisotto and Ruslan Salakhutdinov, ICLR 2018

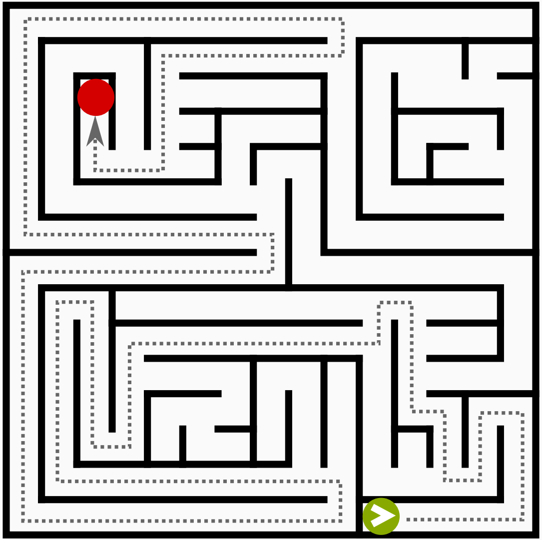
Presented by
Andreas Stöckel

UNIVERSITY OF WATERLOO

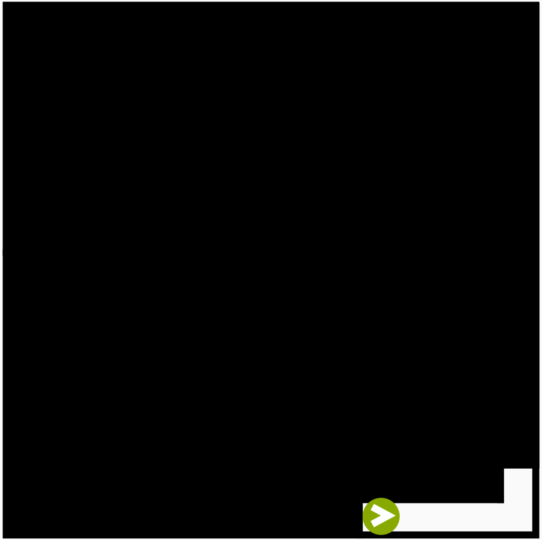
July 6, 2018

MOTIVATION: NAVIGATING PARTIALLY OBSERVABLE ENVIRONMENTS (I)





MOTIVATION: NAVIGATING PARTIALLY OBSERVABLE ENVIRONMENTS (I)



MOTIVATION: NAVIGATING PARTIALLY OBSERVABLE ENVIRONMENTS (I)

Screenshot of “The Elder Scrolls: Arena”, 1994

Reinforcement learning setup

- **State:**
  Partially observable s_t
- **Reward:**
  Sparse reward r, i.e. only when agent reaches goal
- **Actions:**
  Discrete a_t
- **Goal:**
  Find policy $\pi(a \mid s)$ navigating through arbitrary environments

Reinforcement learning setup

- **State:**
  Partially observable s_t
- **Reward:**
  Sparse reward r, i.e. only when agent reaches goal
- **Actions:**
  Discrete a_t
- **Goal:**
  Find policy $\pi(a \mid s)$ navigating through arbitrary environments

Observations

- Partially observable environment mandates *memory*

Reinforcement learning setup

- **State:**
  Partially observable s_t
- **Reward:**
  Sparse reward r, i.e. only when agent reaches goal
- **Actions:**
  Discrete a_t
- **Goal:**
  Find policy $\pi(a \mid s)$ navigating through arbitrary environments

Observations

- Partially observable environment mandates *memory*
- Long-short-term memory (LSTM) units tend to *forget quickly*

Reinforcement learning setup

- **State:**
  Partially observable s_t
- **Reward:**
  Sparse reward r, i.e. only when agent reaches goal
- **Actions:**
  Discrete a_t
- **Goal:**
  Find policy $\pi(a \mid s)$ navigating through arbitrary environments

Observations

- Partially observable environment mandates *memory*
- Long-short-term memory (LSTM) units tend to *forget quickly*
- External memories such as Differentiable Neural Computer (DNC) have problems with *interference*

MOTIVATION: NAVIGATING PARTIALLY OBSERVABLE ENVIRONMENTS (II)

Reinforcement learning setup

- **State:**
  Partially observable s_t
- **Reward:**
  Sparse reward r, i.e. only when agent reaches goal
- **Actions:**
  Discrete a_t
- **Goal:**
  Find policy $\pi(a \mid s)$ navigating through arbitrary environments

Observations

- Partially observable environment mandates *memory*
- Long-short-term memory (LSTM) units tend to *forget quickly*
- External memories such as Differentiable Neural Computer (DNC) have problems with *interference*

$\Rightarrow$ Reduce interference by incorporating *locality* into DNC

OVERVIEW

Part I

Background

EXTERNAL NEURAL NETWORK MEMORIES

No write operator

Memory networks

Convolutional network over past M states

Write operator

Differentiable Neural Computer

External memory matrix with differentiable access operations

- External memory matrix $M \in \mathbb{R}^{W \times H}$
- **Associative memory:**
  - Associate context with value

$$c_t \longrightarrow v_t$$

  - Given $\tilde{c}_t \approx c_t$ retrieve $r_t \approx v_t$

- External memory matrix $M \in \mathbb{R}^{W \times H}$
- **Associative memory:**
  - Associate context with value

$$c_t \rightarrow v_t$$

  - Given $\tilde{c}_t \approx c_t$ retrieve $r_t \approx v_t$

- **Read operation:**
  Given read context vector c_t^{R}

$$r_t = M_t^T c_t^{\mathrm{R}}$$

- **Write operation:**
  Given write context c_t^{W},
  erase vector e_t, value v_t

$$M_{t+1} = M_t \circ (1 - c_t^{\mathrm{W}} e_t^T) + c_t^{\mathrm{W}} v_t^T$$

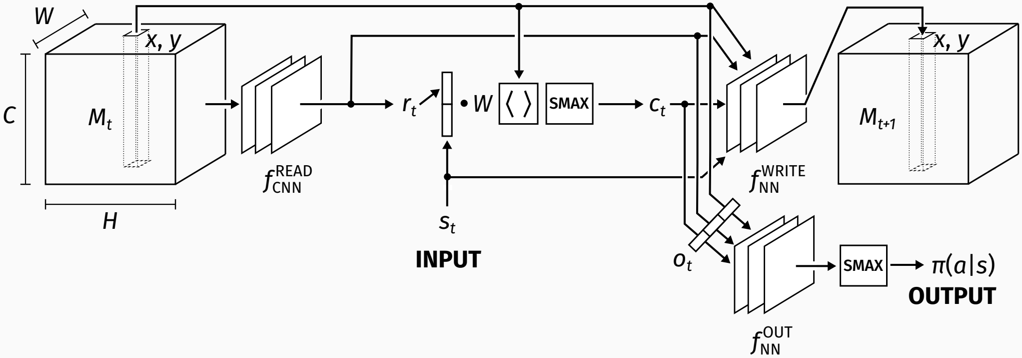
- `REINFORCE` policy gradient descent with value function $V^\pi(s)$ as baseline (Actor-Critic)

$$\nabla_\pi \log \pi(a_t \mid s_t)\big(G_t - V^\pi(s_t)\big)$$
$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

- **Here:** $\pi(a \mid s)$ is deep neural network

Part II

NEURAL MAP NETWORK

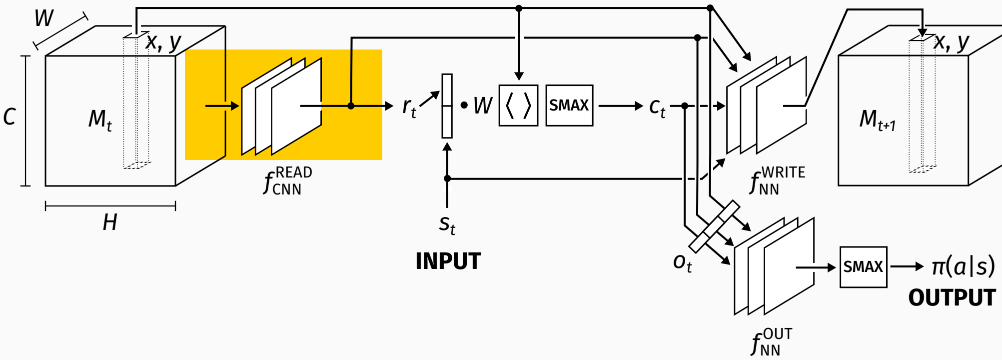

W
x, y
C
M_t
H
$f_{\text{CNN}}^{\text{READ}}$
r_t
• W
⟨ ⟩
SMAX
c_t
s_t
INPUT
$f_{\text{NN}}^{\text{WRITE}}$
M_{t+1}
x, y
o_t
$f_{\text{NN}}^{\text{OUT}}$
SMAX
$\pi(a|s)$
OUTPUT

Variables

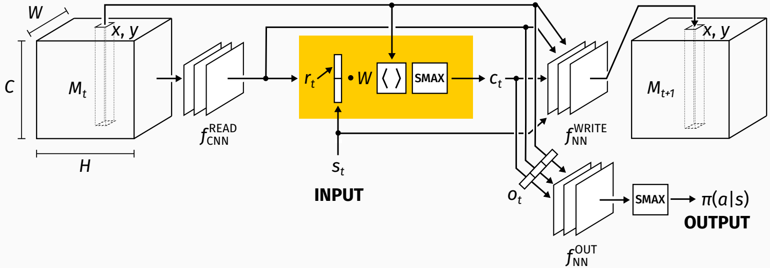
Time step	$t \in \mathbb{Z}$
Location	$(x_t, y_t) \in \mathbb{R}^2$
Input state	$s_t \in \mathbb{R}^n$
Neural map	$M_t \in \mathbb{R}^{C \times H \times W}$

Operations

Read	r_t	$= read(M_t)$
Context	c_t	$= context(M_t, s_t, r_t)$
Write	$w_{t+1}^{(x_t,y_t)}$	$= write\left(s_t, r_t, c_t, M_t^{(x_t,y_t)}\right)$
Update	M_{t+1}	$= update\left(M_t, w_{t+1}^{(x_t,y_t)}\right)$
Output	o_t	$= \left[r_t, c_t, w_{t+1}^{(x_t,y_t)}\right]$
Policy	$\pi(s_t \mid a)$	$= \mathrm{SoftMax}\left(f_{\mathrm{NN}}^{\mathrm{OUT}}(o_t)\right)$

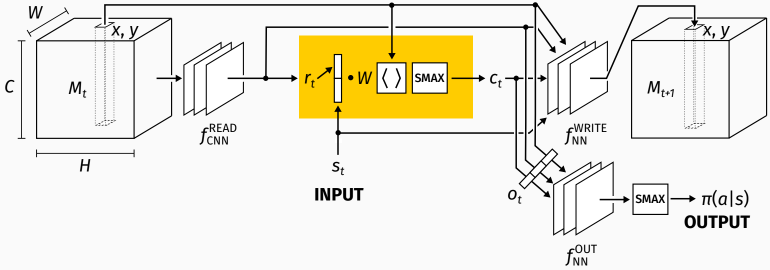


- Summarize memory in single "read vector" $r_t = f_{\text{CNN}}^{\text{READ}}(M_t) \in \mathbb{R}^C$



- Decode context c_t based on input state s_t

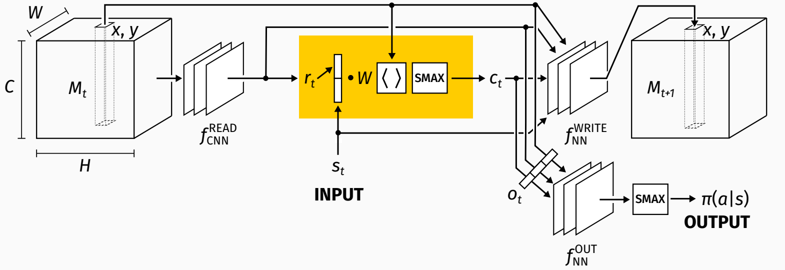
$$q_t = W\left[s_t, r_t\right] \in \mathbb{R}^C$$



- Decode context c_t based on input state s_t

$$q_t = W\left[s_t, r_t\right] \in \mathbb{R}^C$$

$$a_t^{(x,y)} = \left\langle q_t, M_t^{(x,y)} \right\rangle \in \mathbb{R}$$

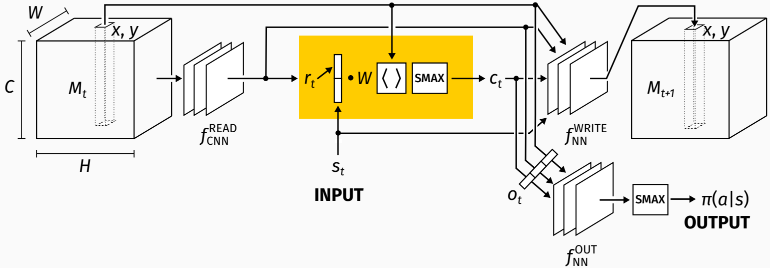


- Decode context c_t based on input state s_t

$$q_t = W\left[s_t, r_t\right] \in \mathbb{R}^C$$

$$a_t^{(x,y)} = \langle q_t, M_t^{(x,y)} \rangle \in \mathbb{R}$$

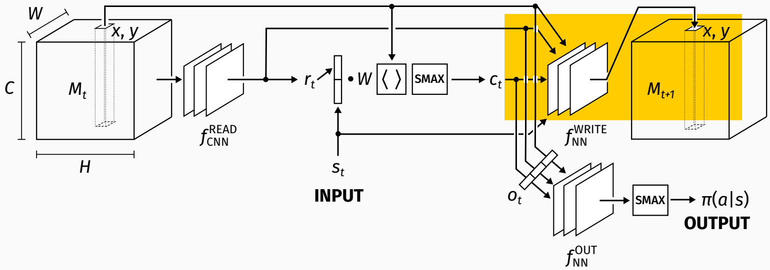
$$\alpha_t^{(x,y)} = \frac{\exp\left(a_t^{(x,y)}\right)}{\sum_{z,w} \exp\left(a_t^{(z,w)}\right)} \in \mathbb{R}$$



- Decode context c_t based on input state s_t

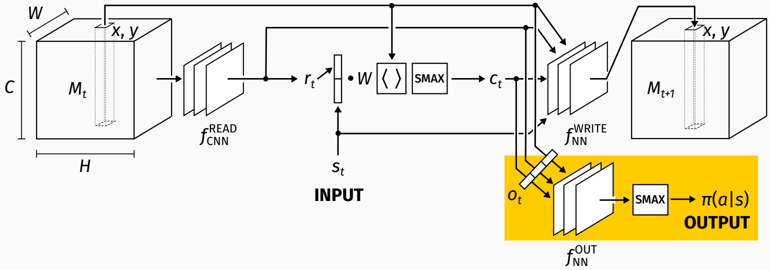
$$q_t = W[s_t, r_t] \in \mathbb{R}^C \qquad \alpha_t^{(x,y)} = \frac{\exp\left(a_t^{(x,y)}\right)}{\sum_{z,w} \exp\left(a_t^{(z,w)}\right)} \in \mathbb{R}$$

$$a_t^{(x,y)} = \langle q_t, M_t^{(x,y)} \rangle \in \mathbb{R} \qquad c_t = \sum_{x,y} \alpha_t^{(x,y)} M_t^{(x,y)} \in \mathbb{R}^C$$



- Compute content of memory for $t+1$ at location x, y

$$w_{t+1}^{(x_t,y_t)} = f_{NN}^{WRITE}\left(\left[s_t, r_t, c_t, M_t^{(x_t,y_t)}\right]\right) \qquad M_{t+1}^{(a,b)} = \begin{cases} w_{t+1}^{(x_t,y_t)} & \text{if } (a,b) = (x_t,y_t) \\ M_{t+1}^{(a,b)} & \text{if } (a,b) \neq (x_t,y_t) \end{cases}$$



- Compute policy $\pi(a \mid s)$

$$o_t = \left[c_t, r_t, M_t^{(x_t, y_t)}\right] \qquad \pi(a \mid s) = \mathrm{SoftMax}\left(f_{\mathrm{NN}}^{\mathrm{OUT}}(o_t)\right)$$

- **Write operation: Gated Recurrent Units (GRUs)**
  Use GRU equations to disable/weaken memory update

- **Write operation: Gated Recurrent Units (GRUs)**
  Use GRU equations to disable/weaken memory update

- **LSTM controller for 3D environments**
  - W, H too small, confuses network (reads what it just wrote)

  $\Rightarrow$ Use LSTM to remember read/write operations

$$h_t = \mathrm{LSTM}(s_t, r_t, c_{t-1}, h_{t-1}) \qquad c_t = context(M_t, h_t)$$

- **Write operation: Gated Recurrent Units (GRUs)**
  Use GRU equations to disable/weaken memory update

- **LSTM controller for 3D environments**
  - W, H too small, confuses network (reads what it just wrote)

  $\Rightarrow$ Use LSTM to remember read/write operations

$$h_t = \mathrm{LSTM}(s_t, r_t, c_{t-1}, h_{t-1}) \qquad\qquad c_t = context(M_t, h_t)$$

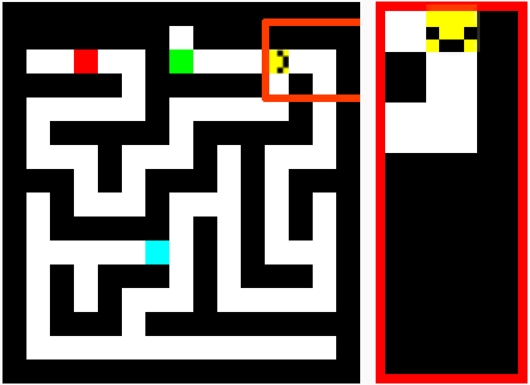
- **Egocentric navigation**
  - Real-world agents do not know their global x, y location

  $\Rightarrow$ Always write to centre of memory, translate memory on movement

Part III

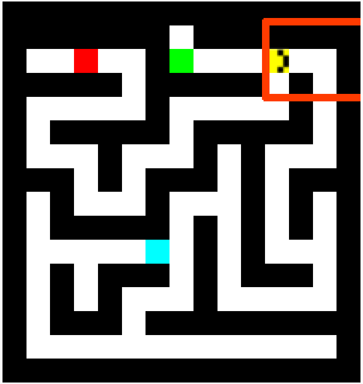
Empirical Evaluation

- **Environment:**
  Randomly generated 2D maze (sizes 7-11, 13-15; 1000 mazes held back for testing)

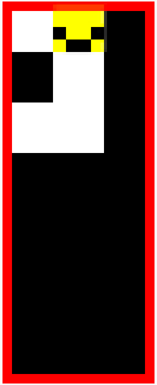


(a) Maze ↑ (b) Observation ↑

- **Environment:**
  Randomly generated 2D maze (sizes 7-11, 13-15; 1000 mazes held back for testing)
- **Task:**
  Indicator selects goal
  Blue indicator → teal goal
  Green indicator → red goal

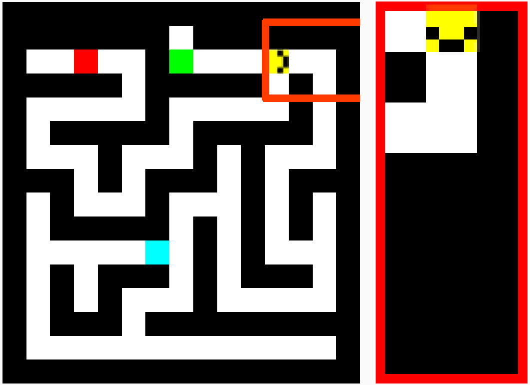


(a) Maze ↑



(b) Observation ↑

- **Environment:**
  Randomly generated 2D maze
  (sizes 7-11, 13-15; 1000 mazes held back for testing)
- **Task:**
  Indicator selects goal
  Blue indicator → teal goal
  Green indicator → red goal
- **Input/output:**
  Input: RGB image, local slice
  Output: $\{\uparrow, \curvearrowleft, \curvearrowright\}$



(a) Maze ↑ (b) Observation ↑

2D GOAL-SEARCH ENVIRONMENT: RESULTS (I)

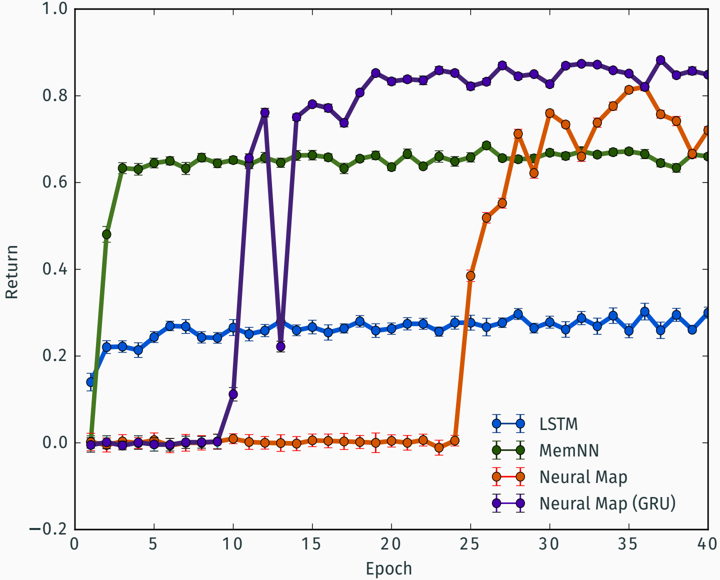


Figure 3: Training curves for all 4 agent architectures on the "Goal-Search" environment. The *x*-axis is an epoch (250k concurrent steps) and the *y*-axis is the average undiscounted episode return. The curves show that the GRU-based Neural Map learns faster and is more stable than the standard update Neural Map.

2D GOAL-SEARCH ENVIRONMENT: RESULTS (II)

	2D GOAL SEARCH					
	TRAIN			TEST		
AGENT	7-11	13-15	TOTAL	7-11	13-15	TOTAL
Random Baseline	41.9%	25.7%	38.1%	46.0%	29.6%	38.8%
LSTM	84.7%	74.1%	87.4%	96.3%	83.4%	91.4%
MQN-32	80.2%	64.4%	83.3%	95.9%	74.6%	87.4%
MQN-64	83.2%	69.6%	85.8%	96.5%	76.7%	88.3%
Neural Map (15x15)	92.4%	80.5%	89.2%	93.5%	87.9%	91.7%
Neural Map + GRU (15x15)	97.0%	89.2%	94.9%	97.7%	94.0%	96.4%
Neural Map + GRU (8x8)	94.9%	90.7%	95.6%	98.0%	95.8%	97.3%
Neural Map + GRU + Pos (8x8)	95.0%	91.0%	95.9%	98.3%	94.3%	96.5%
Neural Map + GRU + Pos (6x6)	90.9%	83.2%	91.8%	97.1%	90.5%	94.0%
Ego Neural Map + GRU (15x15)	94.6%	91.1%	95.4%	97.7%	92.1%	95.5%
Ego Neural Map + GRU + Pos (15x15)	74.6%	63.9%	78.6%	87.8%	73.2%	82.7%

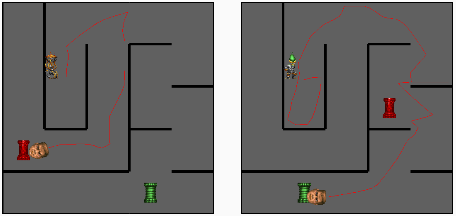
- Memory Size
- Gated Recurrent Unit
- Position Part of State
- Best result in group

Table 1: Results of several different agent architectures on the "Goal-Search" environment. The "train" columns represent the number of mazes solved (in %) when sampling from the same distribution as used during training. The "test"columns represent the number of mazes solved when run on a set of held-out maze samples which are guaranteed not to have been sampled during training.

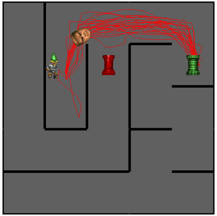
- **Environment:**
  Random 2D maze;
  rendered in 3D
  (10 test mazes)

- **Input/output:**
  Input: RGB image
  Output: $\{\uparrow, \curvearrowleft, \curvearrowright\}$

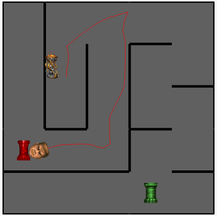
Indicator task



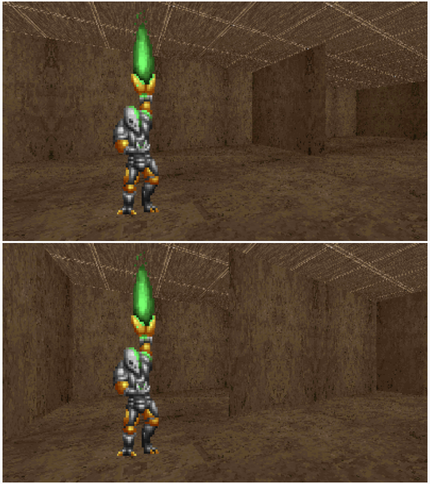
Repeat task



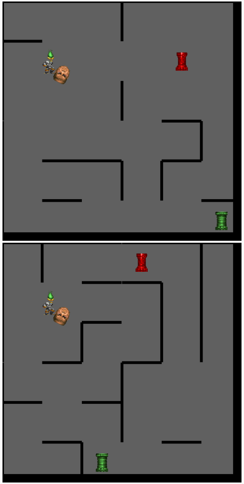
Minotaur task



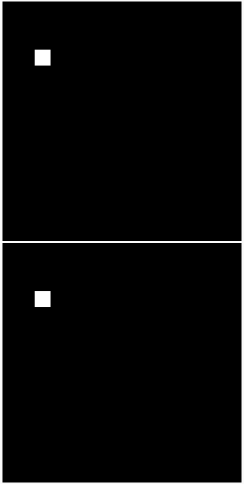
RGB INPUT



MAZE

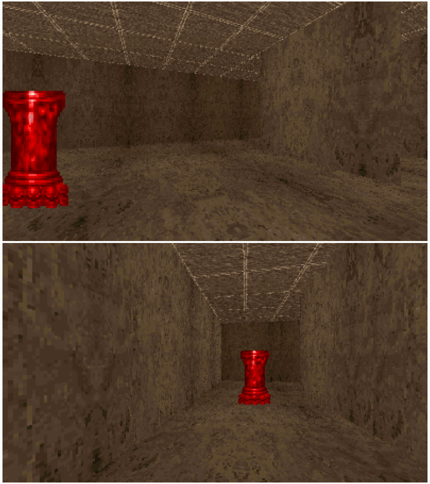


MEMORY α_t

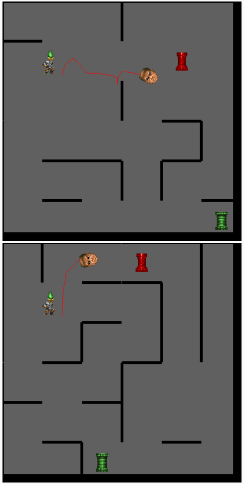


3D DOOM ENVIRONMENT: EXAMPLES (ALLOCENTRIC)

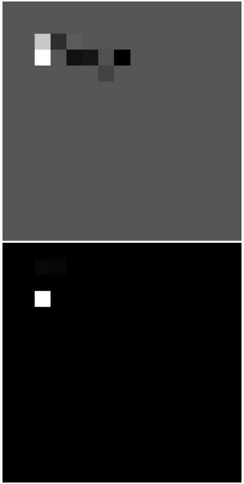
RGB INPUT



MAZE

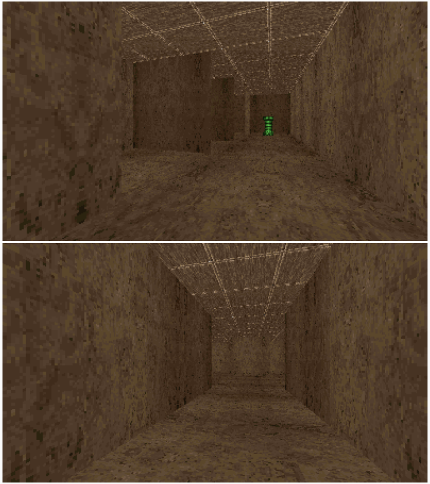


MEMORY α_t

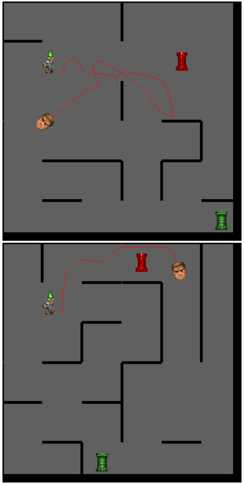


3D DOOM ENVIRONMENT: EXAMPLES (ALLOCENTRIC)

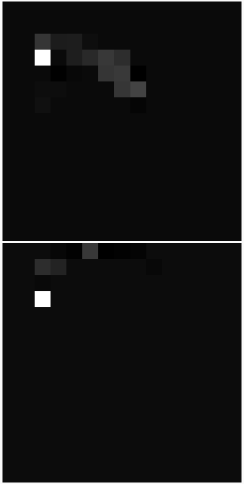
RGB INPUT



MAZE

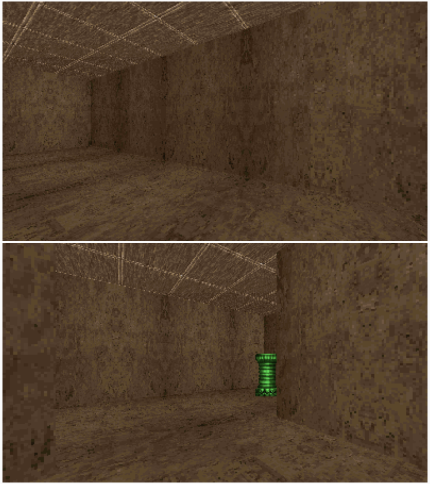


MEMORY α_t

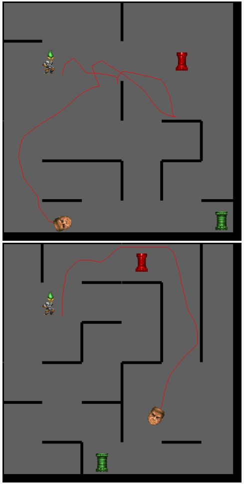


3D DOOM ENVIRONMENT: EXAMPLES (ALLOCENTRIC)

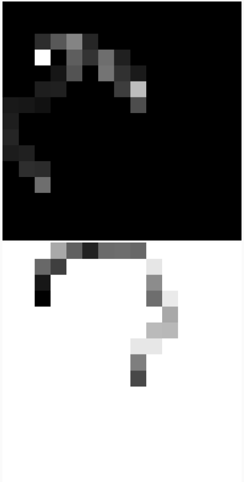
RGB INPUT



MAZE

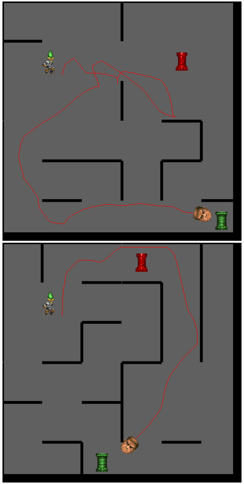


MEMORY α_t

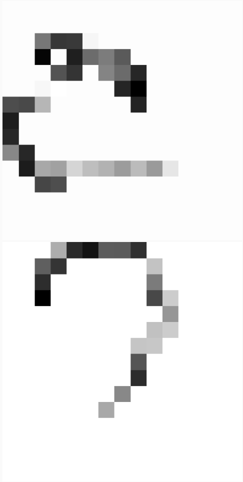


RGB INPUT

MAZE



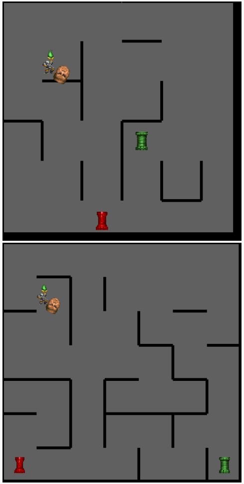
MEMORY α_t



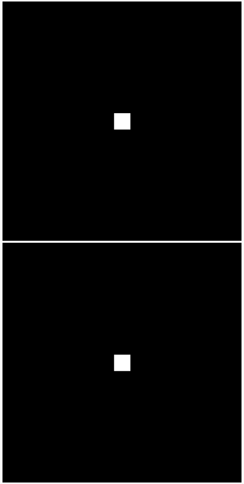
3D DOOM ENVIRONMENT: EXAMPLES (EGOCENTRIC)

RGB INPUT

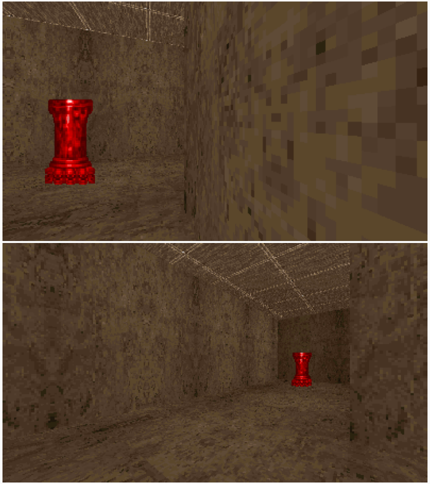
MAZE



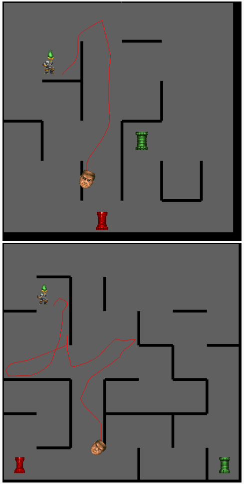
MEMORY α_t



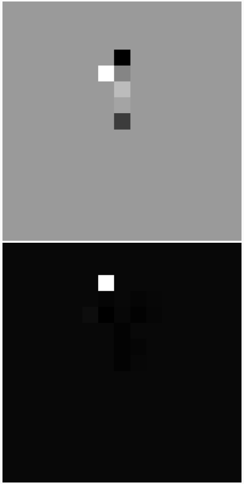
RGB INPUT



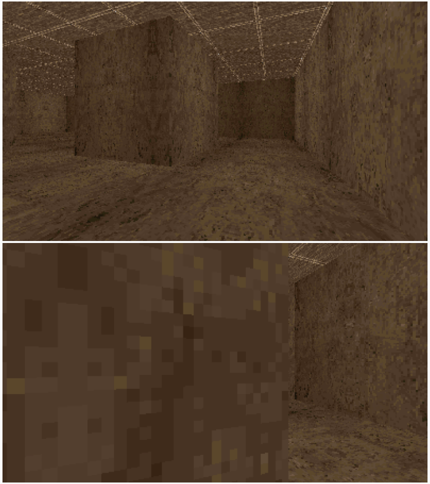
MAZE



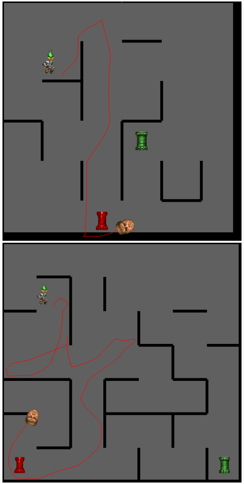
MEMORY α_t



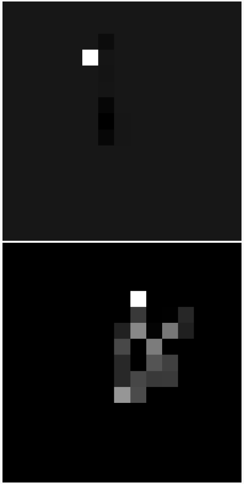
RGB INPUT



MAZE

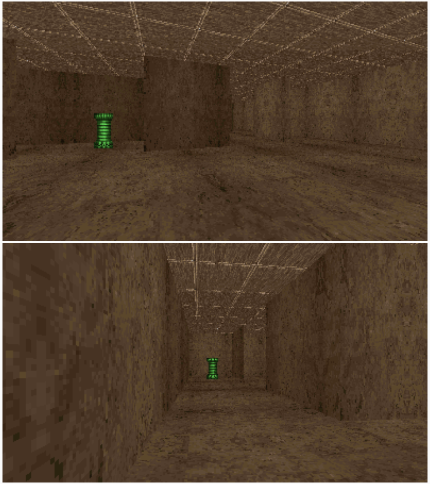


MEMORY α_t

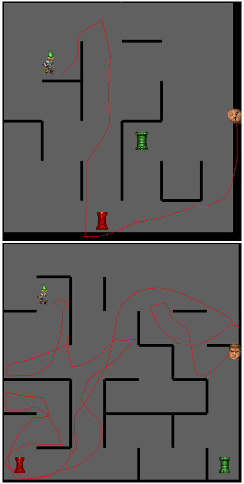


3D DOOM ENVIRONMENT: EXAMPLES (EGOCENTRIC)

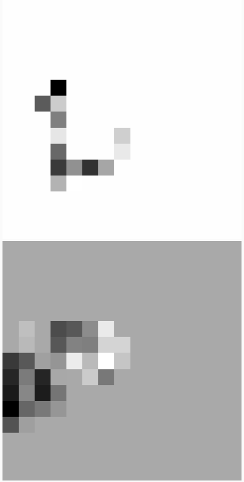
RGB INPUT



MAZE



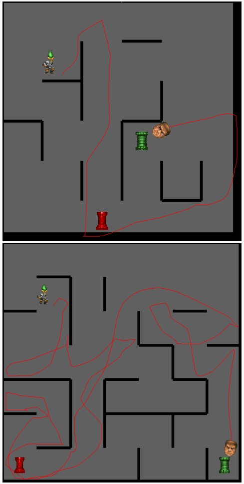
MEMORY α_t



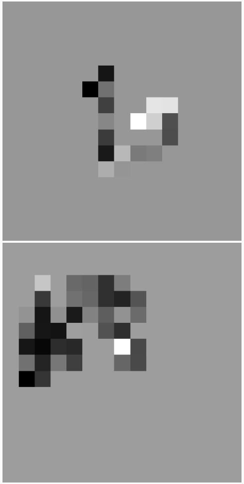
3D DOOM ENVIRONMENT: EXAMPLES (EGOCENTRIC)

RGB INPUT

MAZE



MEMORY α_t



3D DOOM ENVIRONMENT: RESULTS

		3D DOOM ENVIRONMENT — TASK														
	MAZE SIZE	INDICATOR					REPEATING					MINOTAUR				
AGENT		4	5	6	7	8	4	5	6	7	8	4	5	6	7	8
LSTM	ACC	95.7	87.5	81.1	71.4	60.3	-	-	-	-	-	90.0	71.5	48.0	34.2	29.4
	REW	-	-	-	-	-	7.26	7.58	6.06	5.32	4.98	1.35	1.07	0.72	0.51	0.44
FRMQN	ACC	87.3	82.9	78.0	72.0	59.8	-	-	-	-	-	72.7	54.5	38.8	28.8	23.7
	REW	-	-	-	-	-	1.45	1.65	1.51	1.37	1.09	1.09	0.82	0.58	0.43	0.36
Controller Neural Map	ACC	**95.8**	90.3	81.8	80.4	70.3	-	-	-	-	-	**99.7**	**92.2**	**67.5**	37.9	30.2
	REW	-	-	-	-	-	**17.4**	**17.1**	**12.0**	**11.4**	**12.3**	**1.50**	**1.38**	**1.01**	0.57	0.45
Controller Ego Neural Map	ACC	94.6	**91.0**	**87.6**	**85.8**	**72.2**	-	-	-	-	-	98.6	90.0	65.2	**44.7**	**33.8**
	REW	-	-	-	-	-	12.8	14.1	11.0	10.4	9.72	1.48	1.35	0.98	**0.67**	**0.51**

● Accuracy ● Reward ● Best result

Table 2: Accuracy for Indicator means % of correct goals reached, for Minotaur it means % of episodes where the agent successfully reached the goal and backtracked to the beginning. Reward for Repeating is number of times correct goal was visited within the allotted timesteps (+1 for correct goal, -1 for incorrect goal). Reward for Minotaur is +0.5 for reaching the goal and +1.0 for backtracking (max episode reward is +1.5).

Part IV

Summary & Conclusion

- Neural Map is an *extension* to the differentiable neural computer (DNC) that takes *locality* into account

SUMMARY & CONCLUSION

- Neural Map is an *extension* to the differentiable neural computer (DNC) that takes *locality* into account
- Experiments show *superior performance* to prior art in navigation tasks in *partially observable* 2D/3D environments

SUMMARY & CONCLUSION

- Neural Map is an *extension* to the differentiable neural computer (DNC) that takes *locality* into account
- Experiments show *superior performance* to prior art in navigation tasks in *partially observable* 2D/3D environments

  However: surprisingly small gain compared to memory networks in some experiments

- Neural Map is an *extension* to the differentiable neural computer (DNC) that takes *locality* into account
- Experiments show *superior performance* to prior art in navigation tasks in *partially observable* 2D/3D environments

  However: surprisingly small gain compared to memory networks in some experiments
- **Future work:**
  - Demonstrate scalability to *higher-dimensional* (abstract) tasks and *larger* environments
  - Test with stochastic partially observable state (*POMDPs*)
  - Extend concept of locality to other modalities than position

- Neural Map is an *extension* to the differentiable neural computer (DNC) that takes *locality* into account
- Experiments show *superior performance* to prior art in navigation tasks in *partially observable* 2D/3D environments

  However: surprisingly small gain compared to memory networks in some experiments
- **Future work:**
  - Demonstrate scalability to *higher-dimensional* (abstract) tasks and *larger* environments
  - Test with stochastic partially observable state (*POMDPs*)
  - Extend concept of locality to other modalities than position

Thank you for your attention!