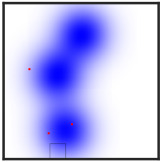
Learning to Search for Targets

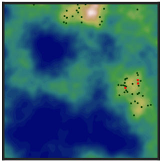
with Deep Reinforcement Learning

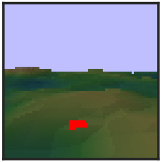
Oskar Lundin

Linköping University

June 10, 2022







Outline

Description

Learned autonomous search for a set of targets in a visual environment with a camera.

- Camera perceives limited region of environment.
- Moving camera changes visible region.
- Detect when targets are visible.
- Locate targets in minimum time.
- Learn control from sample scenarios.
- Use deep reinforcement learning.
- Focus on search behavior, simple detection.

Motivation

- Applications in search and rescue, surveillance, home assistance, etc.
- Autonomous systems may reduce risk and cost.
- Learning vs. handcrafted systems:
  - May find better solutions (deep RL: Atari [1], Go [2], StarCraft II [3]).
  - Applicable as long as data is available.
  - Just describe problem.
  - Guarantees and understandability.

Aim

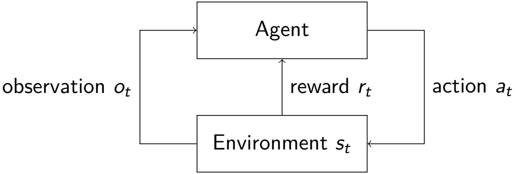
- Utilize structure in environments:
  - Books are in bookshelves, cars on roads. . .
  - Targets can be spread out/close together. . .
- Learn distribution of targets from training samples.
  - Realistically limited training samples available.
  - Generalize to similar unseen search tasks.
- Remember features of explored environment to:
  - Avoid searching regions twice.
  - Prioritize promising regions.

Research Questions

1. How can an agent that learns to intelligently search for targets be implemented with deep reinforcement learning?
2. How does the learning agent compare to random, greedy, exhaustive and human searchers with prior knowledge of the searched scenes?
3. How well does the learning agent generalize from a limited number of training samples to unseen in-distribution search scenarios?

Reinforcement Learning I

- Learn from interaction how to achieve a goal.
- Partially Observable Markov Decision Process [4]:
  - *Agent* interacts with *environment* over discrete time steps $t = 0, 1, 2 \ldots, T$.
  - New state s_{t+1} depends on history $a_0, o_1, r_1, \ldots, a_{t-1}, o_t, r_t$.
  - Agent usually maintains internal state $\rightarrow$ memory.



Reinforcement Learning II

- Policy $\pi(a|s)$ is a mapping from states to action probabilities.
- Find policy that maximizes expected future reward $\mathbb{E}\left[\sum_{k=0}^{T} \gamma^{k-t-1} r_k\right]$.
- Reward signal is often a design parameter.
- Deep reinforcement learning: approximate π with deep neural networks.

Related Work

- Visual attention:
  - Sequential focus points for foveated vision [5].
- Visual navigation:
  - Solve random mazes [6].
  - Find target object in indoor scenes [7].
- Object detection:
  - Region proposals for object localization [8].
  - Contextual reasoning over spatial layout in scenes [9].
  - Anatomical landmark detection in medical images [10].

Problem Statement

- Agent searches scene $S \subset \mathbb{R}^d$.
- Agent perceives view $V \subset S$.
- View can be transformed to new subspace.
- Targets in scene $\{t_0, \ldots t_n\}$, $t_i \in S$.
- Indicate when targets are visible, i.e. $V \cap T \neq \varnothing$.
- Goal:
    - Maximize probability of finding all targets.
    - Minimize cost in time.
    - NP-complete [11]).

Environments

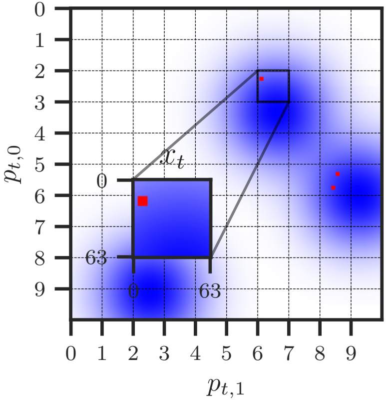
- Three simulated environments.
- Find three targets in less than 1 000 steps.
- Target probability correlated with scene appearance.
- Procedurally generated, conditioned on seed.
- New seed after each finished search.

Observation, Action and Reward

- Observations $o_t = \langle x_t, p_t \rangle$, where
  - $x_t \in \mathbb{R}^{3\times 64\times 64}$ is an RGB image,
  - $p_t \in \{0, \dots, H\} \times \{0, \dots, W\}$ is the camera position.
- Actions $a_t \in \{\texttt{INDICATE}, \texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}\}$, where
  - `INDICATE` identifies targets, and
  - `UP`, `DOWN`, `LEFT`, `RIGHT` move camera.
- Reward $r_t = h - 0.01 + 0.005d + 0.005e$ where
  - $h = |T \cap V|$ if $a_t = \texttt{INDICATE}$, else 0.
  - $d = 1$ if a_t moves closer to nearest target, else 0.
  - $e = 1$ if a_t moves to new position, else 0.

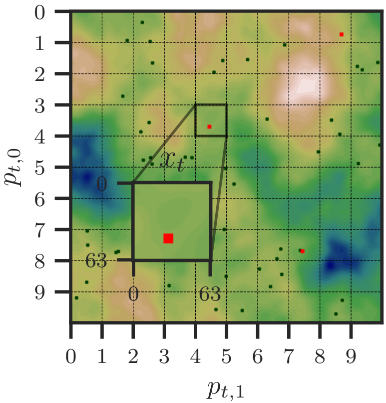
Environment I: Gaussian

- Three gaussian kernels with random center.
- Sum of kernels = blue color intensity, probability of targets.
- Agent should prioritize blue regions.



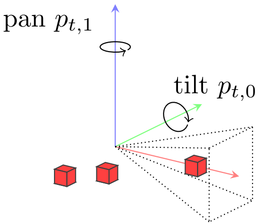
Environment II: Terrain

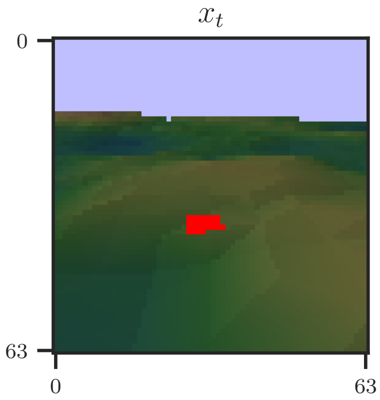
- Terrain seen from above (e.g. UAV).
- Targets between ocean and mountains.
- More realistic, higher variance.



Environment III: Camera

- Terrain seen from perspective projection camera.
- Variance in target appearance.
- Moving actions control pan and tilt.
  - 20 pan angle steps.
  - 10 tilt angle steps.

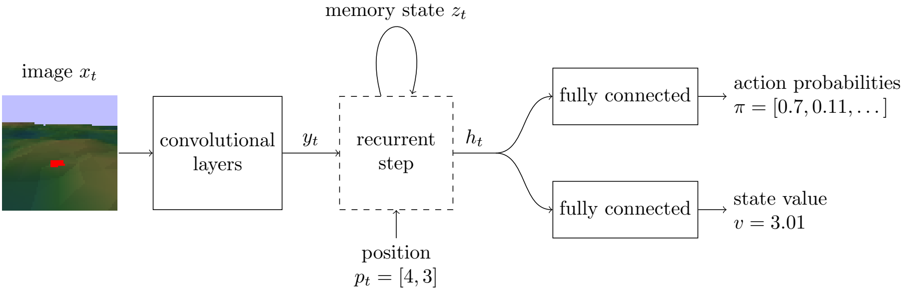




Approach

- Function approximation with deep neural networks.
  - Policy $\pi(a|s, \theta)$.
  - Value $v_\pi(s, \theta)$ (predicts future reward).
- Training procedure:
  1. Collect interactions with environment.
  2. Compute loss $\mathcal{L}(\theta)$.
  3. Optimize $\mathcal{L}$ wrt θ.
  4. Repeat...
- Use proximal policy optimization [12].
  - Clipping loss function $\mathcal{L}_{\text{clip}}$.
  - RL algorithm from 2017.
  - Stable performance, relatively little tuning [13].

Architecture



Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:
  1. Temporal memory (long short-term memory [14]):
     - Previously applied to tasks where memory is required [15, 16, 6, 17].
     - How long sequences can be remembered?
  2. Spatial memory (inspired by [18]):
     - Feature map with one slot per camera position.
     - Indexed with current position.
     - Stores image representation at each slot.
     - Read whole memory with convolutional layers.

Training

- Train for 25M time steps.
- Results reported across 3 training runs.
- Separate training and test sets.
- Same hyperparameters in all runs.

Implementation

- OpenAI Gym environment interface.
- Custom PPO implementation.
- PyTorch for models and automatic differentiation.
- Intel Core i9-10900X CPU.
- NVIDIA GeForce RTX 2080 Ti GPU.

Experiment I: Search Performance

- Compare to simple reference behaviors (baselines).
- Test on held out environment samples.
- Metrics:
  1. Average search path length.
  2. Average success rate.
  3. Success weighted by inverse path length (SPL) [19].

Definition

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)}$$

where N is number of test samples, S_i is binary success indicator, p_i is the taken path length l_i is the shortest path length.

Baselines

Random: randomly samples actions.

Greedy: greedily selects exploring actions (random if none).

Exhaustive: exhaustively covers search space with minimal revisits.

Human: human searcher with knowledge of environment.

Handcrafted: prioritize actions that lead to higher blue intensity (gaussian environment only).

Gaussian Environment

Agent	SPL	Success	Length
random	0.06 ± 0.01	0.92 ± 0.06	369.07 ± 24.93
greedy	0.17 ± 0.00	1.00 ± 0.00	147.12 ± 2.38
exhaustive	0.21 ± 0.00	1.00 ± 0.00	83.37 ± 2.88
handcrafted	0.33 ± 0.00	1.00 ± 0.00	65.20 ± 1.41
human	0.23 ± 0.03	1.00 ± 0.00	80.97 ± 13.49
temporal	0.24 ± 0.03	0.99 ± 0.01	101.25 ± 13.32
spatial	0.29 ± 0.02	0.99 ± 0.01	72.16 ± 5.97

`video 1`, `video 2`, `video 3`.

Terrain Environment

Agent	SPL	Success	Length
random	0.06 ± 0.01	0.89 ± 0.04	366.05 ± 26.96
greedy	0.17 ± 0.01	1.00 ± 0.00	141.01 ± 2.31
exhaustive	0.22 ± 0.00	1.00 ± 0.00	84.11 ± 0.84
human	0.26 ± 0.02	1.00 ± 0.00	76.73 ± 5.33
temporal	0.25 ± 0.02	1.00 ± 0.01	103.76 ± 11.69
spatial	0.27 ± 0.01	1.00 ± 0.00	79.60 ± 6.88

`video 1`, `video 2`, `video 3`.

Camera Environment

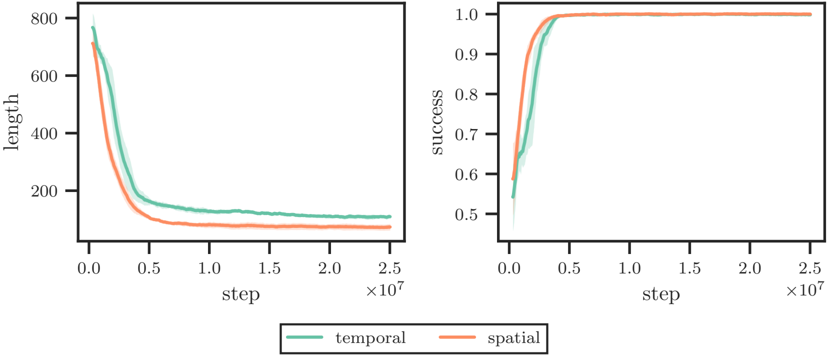
Agent	SPL	Success	Length
random	0.04 ± 0.00	0.62 ± 0.03	545.09 ± 56.25
greedy	0.12 ± 0.01	0.97 ± 0.01	255.60 ± 10.44
exhaustive	0.37 ± 0.00	1.00 ± 0.00	67.03 ± 0.00
human	0.68 ± 0.08	1.00 ± 0.00	38.10 ± 5.72
temporal	0.70 ± 0.02	1.00 ± 0.00	42.36 ± 2.05
spatial	0.66 ± 0.03	1.00 ± 0.00	42.90 ± 1.73

`video 1`, `video 2`, `video 3`.

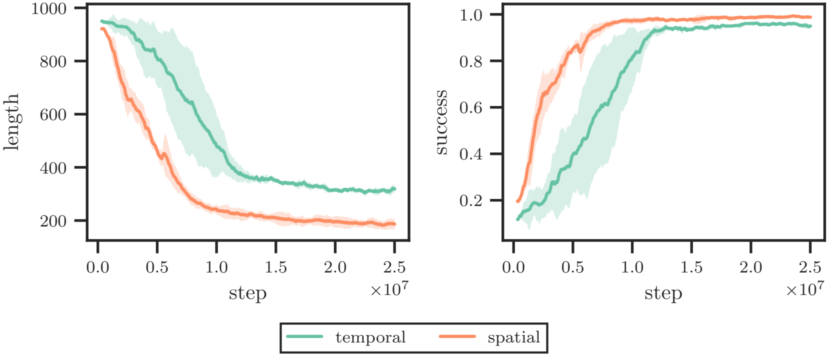
Experiment II: Scaling to Larger Search Spaces

- Larger search spaces are more difficult.
- Stronger demands on memory:
  - Remember visited positions.
  - Remember appearance of environment.
- Compare memories on 10×10, 15×15, and 20×20 versions of gaussian environment.

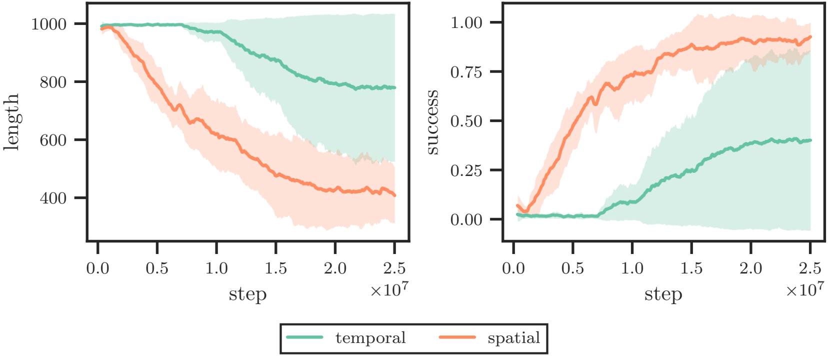
10×10


length
800
600
400
200
0.0
0.5
1.0
1.5
2.0
2.5
step
×10^7
success
1.0
0.9
0.8
0.7
0.6
0.5
temporal
spatial

15×15


1000
800
600
400
200
length
0.0
0.5
1.0
1.5
2.0
2.5
step
×10^7
1.0
0.8
0.6
0.4
0.2
success
step
×10^7
temporal
spatial

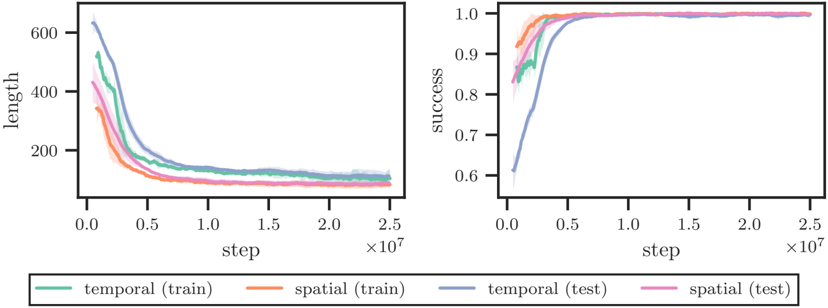
20×20


length
1000
800
600
400
0.0
0.5
1.0
1.5
2.0
2.5
step
×10^7
success
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0
2.5
step
×10^7
temporal
spatial

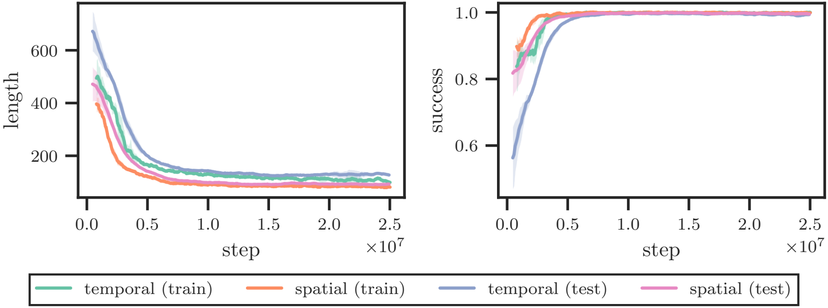
Experiment III: Generalization From Limited Samples

- Real-world tasks usually have limited training samples.
- Train on 500, 1 000, 5 000 and 10 000 samples of terrain environment.
- Test on held out samples from full distribution.

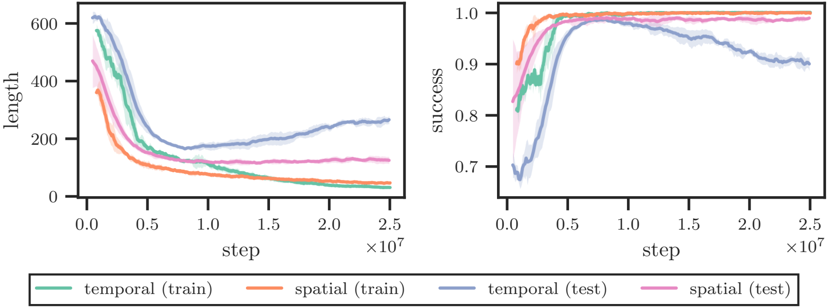
10000 samples



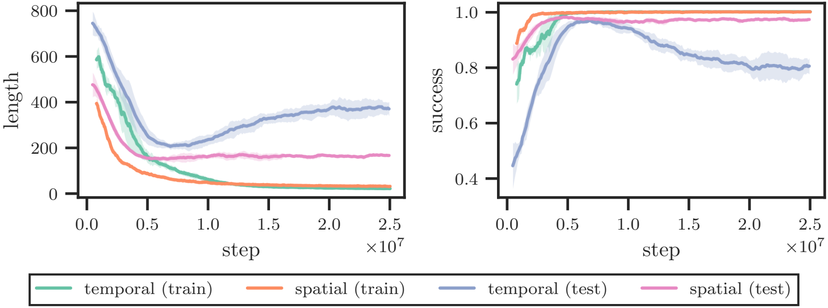
5000 samples



1000 samples



500 samples


length
800
600
400
200
0
0.0
0.5
1.0
1.5
2.0
2.5
step
$\times 10^7$
success
1.0
0.8
0.6
0.4
0.0
0.5
1.0
1.5
2.0
2.5
step
$\times 10^7$
temporal (train)
spatial (train)
temporal (test)
spatial (test)

Conclusion

- General method for visual search with reinforcement learning.
- Three environments for evaluating visual search agents.
- Two different neural network architectures.
- Architecture affects performance, scaling and generalization.
- One approach comparable to human performance.
- Different architectures good for different purposes:
  - Temporal memory (long short-term memory) better for smaller, reactive, search spaces.
  - Spatial memory (structured feature map) better for larger search spaces, generalizes better.
- The approach can be used for visual search
  - Comparable performance to that of a human.
  - Not optimal (worse than handcrafted in at least one environment)

Future Work

- Real-world scenarios.
- Closer to optimal behaviors.
- Formal verification (for security-critical applications).
- Hyperparameter tuning (expensive!).
- Other RL algorithms.

References I

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," vol. 518, no. 7540, pp. 529–533. Number: 7540 Publisher: Nature Publishing Group.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," vol. 529, no. 7587, pp. 484–489. Number: 7587 Publisher: Nature Publishing Group.

[3] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," vol. 575, no. 7782, pp. 350–354. Number: 7782 Publisher: Nature Publishing Group.

References II

[4] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," vol. 101, no. 1, pp. 99–134.

[5] V. Mnih, N. Heess, A. Graves, and k. kavukcuoglu, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc.

[6] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments,"

[7] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3357–3364. 474 citations (Crossref) [2022-05-19].

[8] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning,"

[9] X. Chen and A. Gupta, "Spatial memory for context reasoning in object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4106–4116, IEEE.

References III

[10] F.-C. Ghesu, B. Georgescu, Y. Zheng, S. Grbic, A. Maier, J. Hornegger, and D. Comaniciu, "Multi-scale deep reinforcement learning for real-time 3d-landmark detection in CT scans," vol. 41, no. 1, pp. 176–189.

[11] A. Andreopoulos and J. K. Tsotsos, "A theory of active object localization," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 903–910.
15 citations (Crossref) [2022-05-19] ISSN: 2380-7504.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms,"

[13] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," vol. 32, no. 1.
Number: 1.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," vol. 9, no. 8, pp. 1735–1780.
Conference Name: Neural Computation.

[15] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable MDPs,"

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning,"

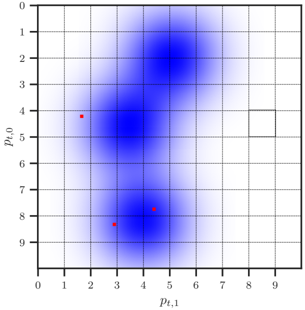
References IV

[17] S. Gupta, V. Tolani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation,"

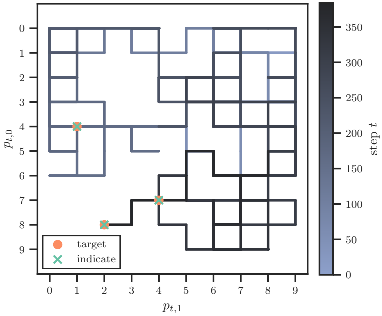
[18] E. Parisotto and R. Salakhutdinov, "Neural map: Structured memory for deep reinforcement learning,"

[19] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents,"

Search Paths I

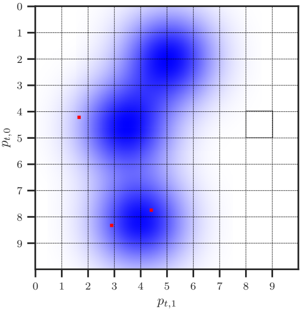


Environment sample

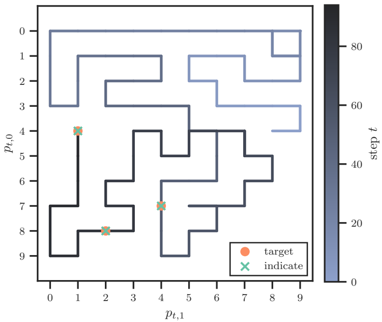


Random baseline

Search Paths II

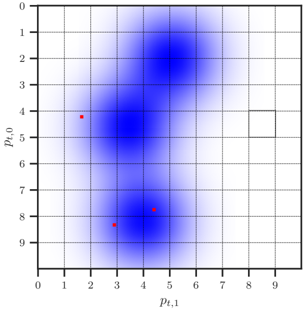


Environment sample

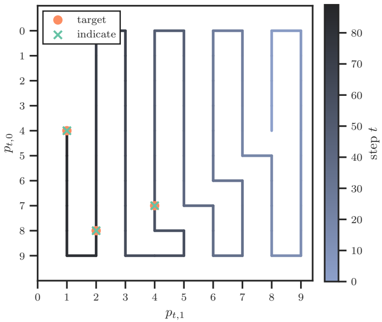


Greedy baseline

Search Paths III

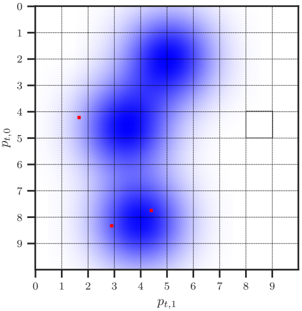


Environment sample

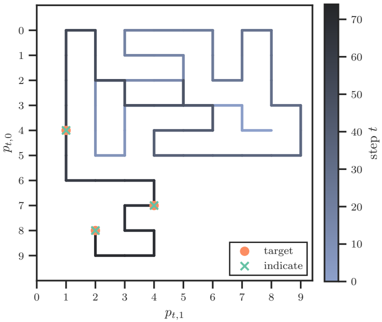


Exhaustive baseline

Search Paths IV

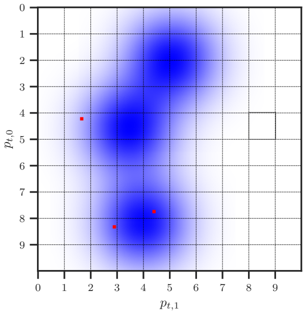


Environment sample

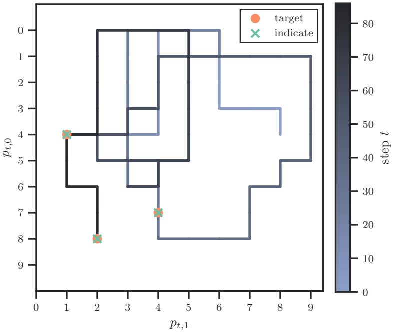


Handcrafted baseline

Search Paths V

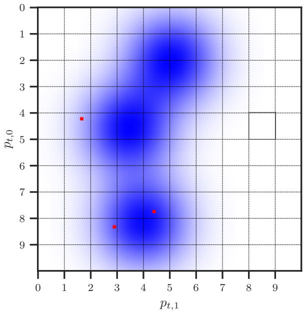


Environment sample

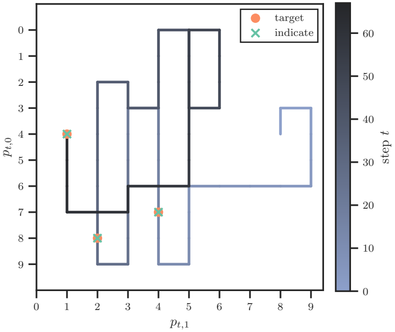


Temporal memory

Search Paths VI

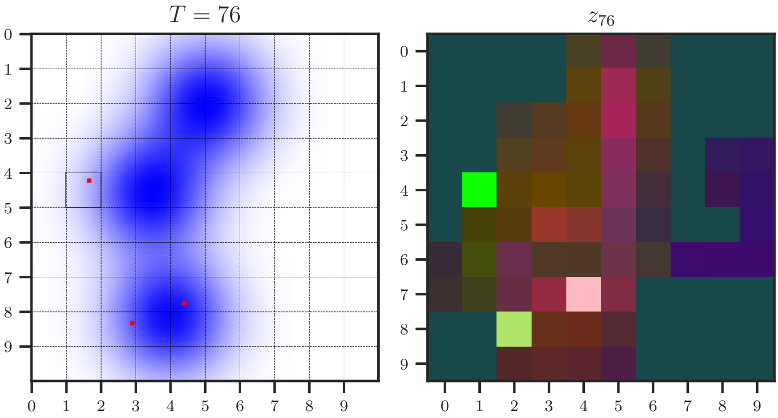


Environment sample



Spatial memory

Memory Viualization



PCA decomposition of spatial memory after episode.