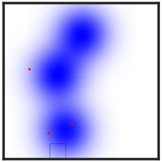
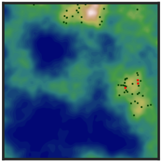
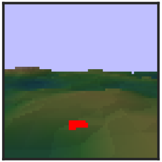
Learning to Search for Targets

with Deep Reinforcement Learning

Oskar Lundin

Linköping University

June 7, 2022

Outline

Description

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

- Camera views limited region of environment.
- Moving camera changes visible region.
- Detect when targets are visible.
- Locate targets in minimum time.
- Utilize visual cues to find targets quicker.
- Learn control from a set of sample scenarios.

Motivation

- Autonomous systems may reduce risks and cost.
- Applications in search and rescue, surveillance, home assistance, etc.
- Handcrafted systems difficult to design.
- Subtle patterns and difficult planning decisions.
- Learning system applicable as long as data is available.

Motivation

- Random or exhaustive search sufficient in small or random environments.
- Most real-world search tasks exhibit structure.
- Visual cues can be used to find targets quicker.
  - Books are in bookshelves.
  - Cars can be found on roads.
  - Some targets spread out/close together.
- Patterns and cues may be subtle and difficult to pick up.
- Manually engineering a searching system with domain knowledge be difficult and costly.

Design Challenges

- Prioritize regions with high probability of targets based on previous experience.
- Learn correlations between scene appearance and target probability.
- Search exhaustively while avoiding searching the same region twice.
- Remember features of searched regions (avoid revisits, scene understanding).
- Real-world tasks have limited number of training samples.

Research Questions

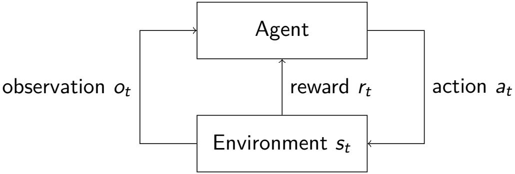
1. How can an agent that learns to intelligently search for targets be implemented with deep reinforcement learning?
2. How does the learning agent compare to random walk, exhaustive search, and a human searcher with prior knowledge of the searched scenes?
3. How well does the learning agent generalize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

Partially Observable Markov Decision Process (POMDP)

POMDP [1]: *Agent* interacts with *environment* over discrete time steps.
At each step $t = 0, 1, 2 \ldots, T$:

- Environment is in state s_t that can not be observed by agent.
- Agent takes *action* a_t.
- Perceives partial *observation* o_t of state.
- Receives scalar reward r_t that indicates whether action is good or bad.
- New state s_{t+1} depends only on history of interactions.
- State not available to agent, must maintain internal state $\rightarrow$ memory.

Goal: select actions that maximizes discounted cumulative reward
$\mathbb{E}\left[\sum_{k=0}^{T} \gamma^{k-t-1} r_k\right]$.



Reinforcement Learning (RL)

RL [2]: Paradigm for learning from interactions how to achieve a goal.

- Policy $\pi(a|s)$ is a mapping from states to action probabilities.
- Find policy that maximizes cumulative reward by interacting with environment.
- Policy gradient methods learn parametrized policy $\pi(a|s, \theta)$ by updating it with some loss function $\mathcal{L}(\theta)$.
- Estimating the value $v_\pi(s, \theta')$ of a state under policy π can be useful to update policy.

Deep RL: Approximate parameters θ of π (and θ' of v_π) with deep neural networks. Has been used to play Atari [3], Go [4], StarCraft II [5], etc.

Proximal Policy Optimization

...

Related Work

Deep RL for similar tasks:

- Visual attention:
  - Sequential focus points for foveated vision [6].
- Visual navigation:
  - Solve random mazes [7].
  - Find target object in indoor scenes [8].
- Object detection:
  - Region proposals for object localization [9].
  - Contextual reasoning over spatial layout in scenes [10].
  - Anatomical landmark detection in medical images [11].

Missing: how visual cues can guide search, overfitting and generalization from limited samples, rigorous performance evaluation.

Problem Statement

- Agent searches scene $S \subset \mathbb{R}^d$.
- Scene contains set of targets $\{t_0, \dots t_n\}$, $t_i \in S$.
- Agent perceives view $V \subset S$.
- View can be transformed to new subspace.
- Indicate when targets are visible, i.e. $V \cup T \neq \varnothing$.
- Maximize the probability of finding all targets while minimizing cost in time (NP-complete [12]).

Environments

- Three simulated environments used for experiments.
- Search space discretized into $H \times W$ camera positions.
- Each camera position has a unique view $V \subset S$.
- Three targets in all scenes.
- Target probability correlated with scene appearance.
- Should be possible to do better than exhaustive search on average.
- Scenes procedurally generated:
  - Pseudorandom seed determines scene appearance and target positions.
  - Gives control over difficulty to solve.
  - Can vary training and test set sizes by limiting seed pool.

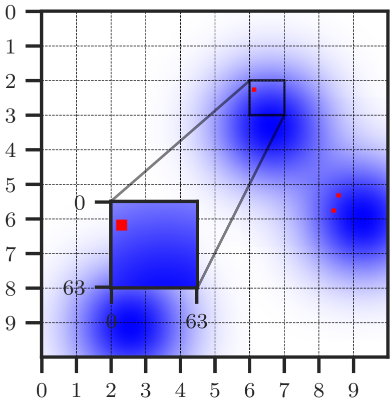
Observation, Action and Reward

At each time step t:

- The agent receives observation $o_t = \langle x_t, p_t \rangle$, where
  - $x_t \in \mathbb{R}^{3\times 64\times 64}$ is an RGB image of current view, and
  - $p_t \in \{0, \ldots, H-1\} \times \{0, \ldots, W-1\}$ is the position of the camera.
- Takes action $a_t \in \{\texttt{INDICATE}, \texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}\}$, where
  - `INDICATE` indicates that a target is in view, and
  - `UP`, `DOWN`, `LEFT`, `RIGHT` move the view in each cardinal direction.
- Receives reward $r_t = h - 0.01 + 0.005d + 0.005e$ where
  - $h = |T \cap V|$ if $a_t = \texttt{INDICATE}$.
  - $d = 1$ if a_t moves closer to nearest target.
  - $e = 1$ if a_t moves to new position.

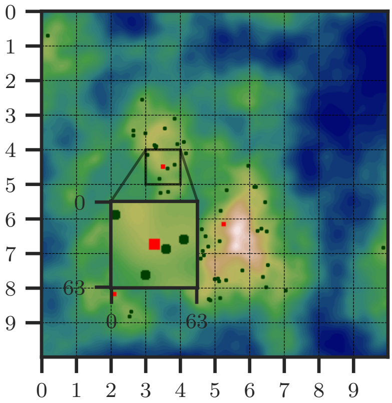
Environment I: Gaussian

- 2D scene, 10×10 search space.
- Three gaussian kernels with random center.
- Sum of kernels determine blue color intensity and probability of targets.
- Clear correlation between appearance and desired behavior.
- Agent should prioritize blue regions.



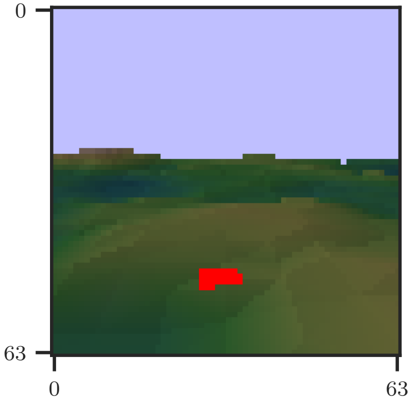
Environment II: Terrain

- Similar to previous environment.
- Terrain seen from above.
- Gradient noise used to generate height map.
- Color determined by height.
- Targets placed with uniform probability across coastlines.
- More realistic, higher variance.
- Analogous to search and rescue with UAV.



Environment III: Camera

- 3D scene viewed from a perspective projection camera.
- Height map from terrain environment turned into mesh, same appearance and target probability as before.
- Camera location fixed at center of scene.
- Moving actions control pan and tilt (pitch and yaw).
- Visually complex, difficult to interpret.

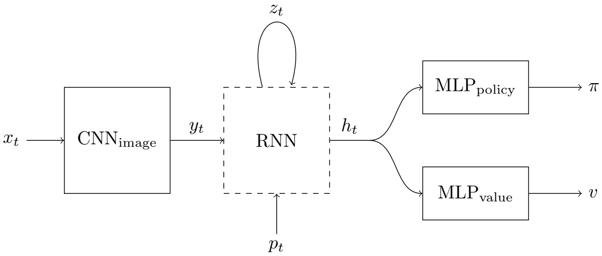


Approach

- Deep reinforcement learning.
- Agent should remember visual features and associate them with their spatial location.

Architecture

- Actor-critic method trained with PPO [13].
- Image x_t passed through CNN.
- Latent image representation y_t and position p_t passed through RNN with recurrent state z_t.
- Policy and value heads approximate π and v_π from h_t with MLPs.



Memory

- Recurrent step serves as memory.
- Necessary to remember features of explored environment.
- Two variants evaluated:
    1. Temporal memory (LSTM)
    2. Spatial memory.

1. LSTM:
   - Proven to work for POMDPs [14, 15, 7, 16].
   - May struggle with remembering over many time steps.
   - Important for exhaustive search and scene understanding.
2. Spatial memory (inspired by [17]):
   - Structured memory $M_t \in \mathbb{R}^{C \times 10 \times 10}$ as hidden state (one slot per camera position p_t / unique view V / image x_t).
   - Read vector $r_t = f(M_t)$, f is CNN.
   - Write vector $w_t = g([h_t, r_t])$, g is MLP.
   - Action probabilities $\pi([r_t, w_t])$ and value $v([r_t, w_t])$.
   - r_t contains information from the whole explored scene.
   - w_t written to index p_t of M_{t+1}.

Experiments

1. Search Performance
2. Scaling to Larger Search Spaces
3. Generalization from Limited Samples

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

Baselines

- Simple handcrafted policies.
- Give a sense of the performance achieved by learning agents.
- All indicate automatically when target visible.
- Random: randomly samples actions.
- Greedy: greedily selects actions lead to unvisited positions (random if none).
- Exhaustive: exhaustively covers search space with minimal revisits.
- Human: human searcher with prior knowledge of environment characteristics.
- Handcrafted (gaussian environment): prioritize actions that lead to higher blue intensity.

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

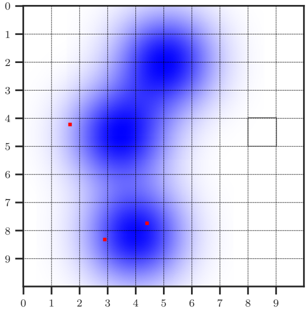
- Compare to baselines.
- Use held out samples as test set.
- Average number of steps on test set.
- SPL metric [18], with N as the number of test samples, S_i indicating success, p_i as the number of steps and l_i as the shortest path length:

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

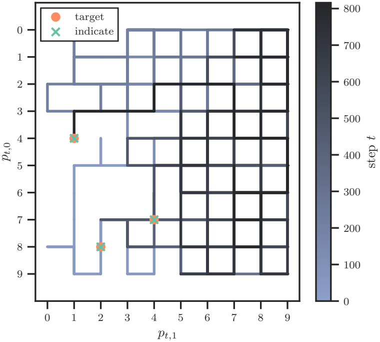
Gaussian Environment

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

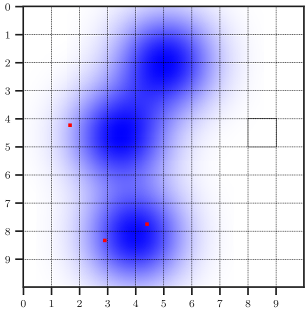
video 1, video 2, video 3 (spatial)



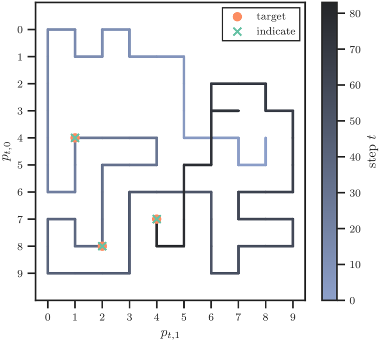
Environment sample



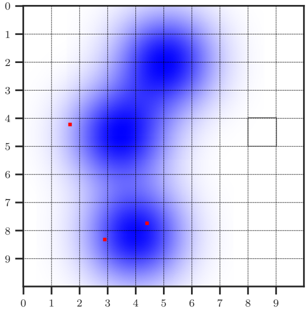
Random baseline



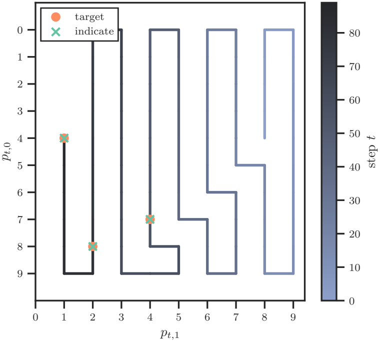
Environment sample



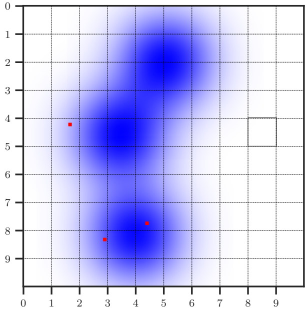
Greedy baseline



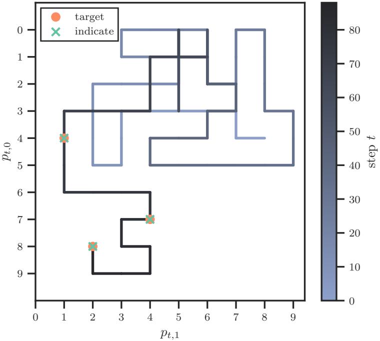
Environment sample



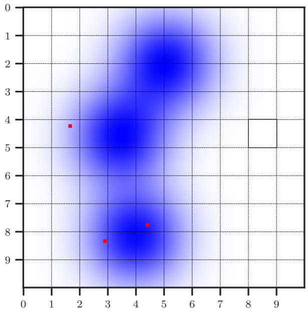
Exhaustive baseline



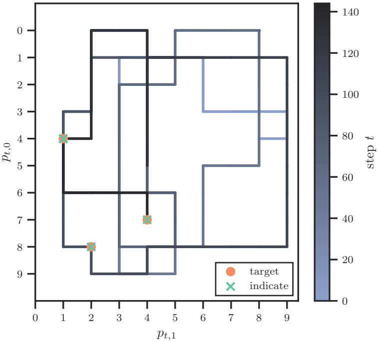
Environment sample



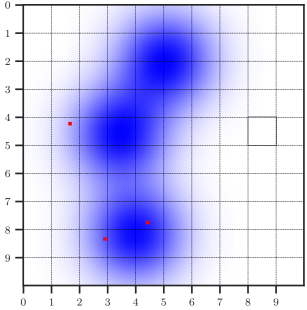
Handcrafted baseline



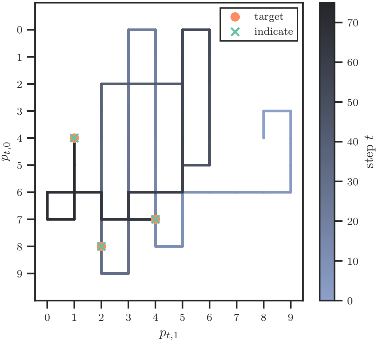
Environment sample



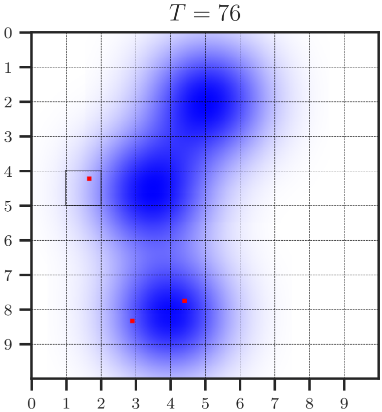
Temporal memory

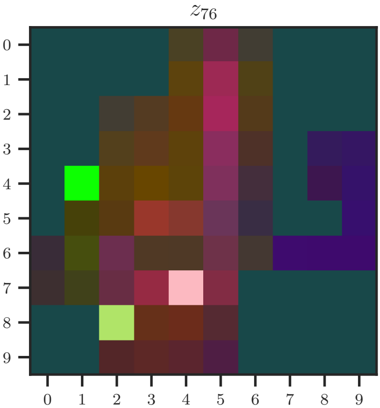


Environment sample



Spatial memory





PCA decomposition of spatial memory after episode.

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 (spatial)

Camera Environment

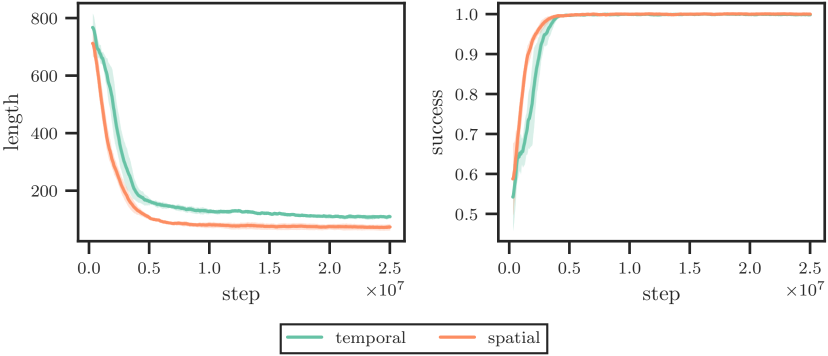
Agent	SPL	Success	Length
random	0.04 ± 0.00	0.62 ± 0.03	545.09 ± 56.25
greedy	0.12 ± 0.01	0.97 ± 0.01	255.60 ± 10.44
exhaustive	0.37 ± 0.00	1.00 ± 0.00	67.03 ± 0.00
human	0.68 ± 0.08	1.00 ± 0.00	38.10 ± 5.72
temporal	0.70 ± 0.02	1.00 ± 0.00	42.36 ± 2.05
spatial	0.66 ± 0.03	1.00 ± 0.00	42.90 ± 1.73

video 1, video 2, video 3 (temporal)

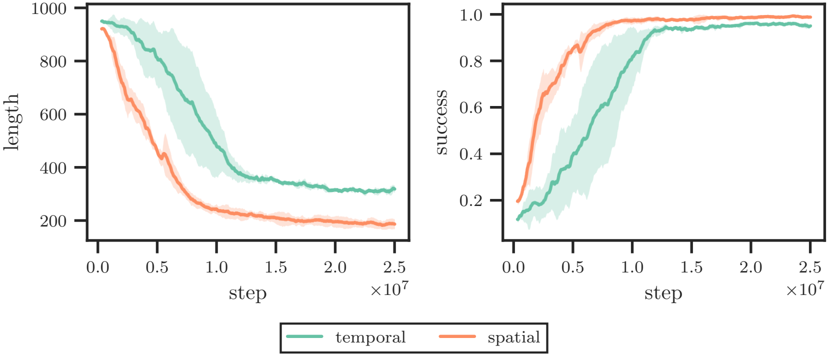
Experiment II: Scaling to Larger Search Spaces

- Larger search spaces take longer to train:
  - More states to explore and exploit.
  - Stronger demands on memory (remember searched positions, scene understanding).
- Investigate impact by comparing agents on 10×10, 15×15, and 20×20 versions of gaussian environment.

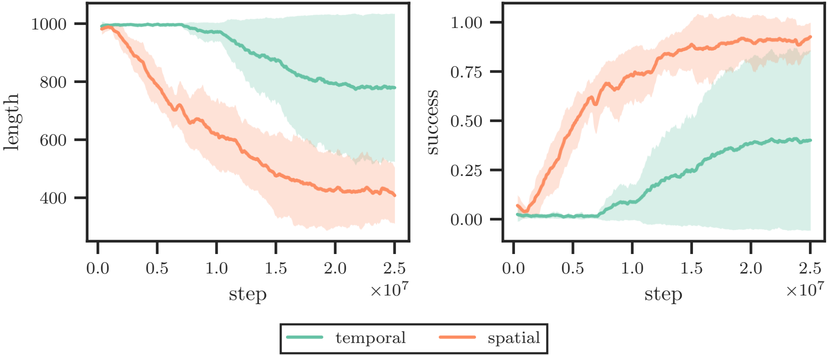
10×10


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

15×15


length
1000
800
600
400
200
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
0.2
temporal
spatial

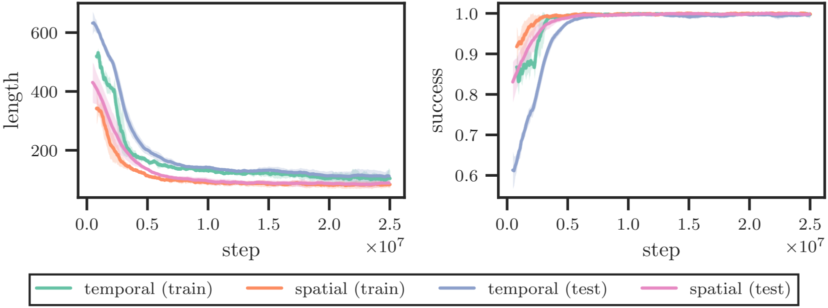
20×20



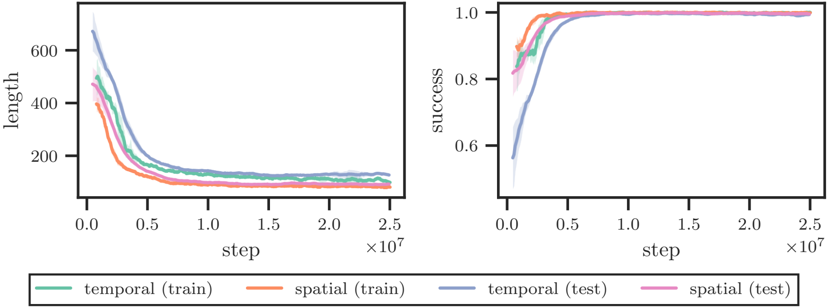
Experiment III: Generalization From Limited Samples

- Limit number of scene samples seen during training to 500, 1 000, 5 000, 10 000.
- Test on held out scenes from full distribution.
- Use terrain environment, high appearance variance and somewhat realistic.
- Fix seed pool used to generate scenes seen during training.
- Train agents until convergence (or for a fixed number of time steps).

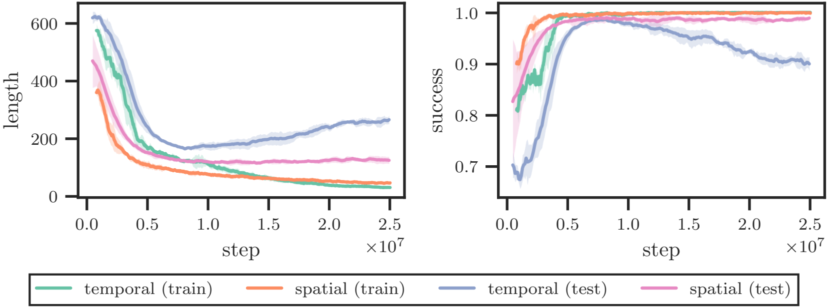
10000 samples



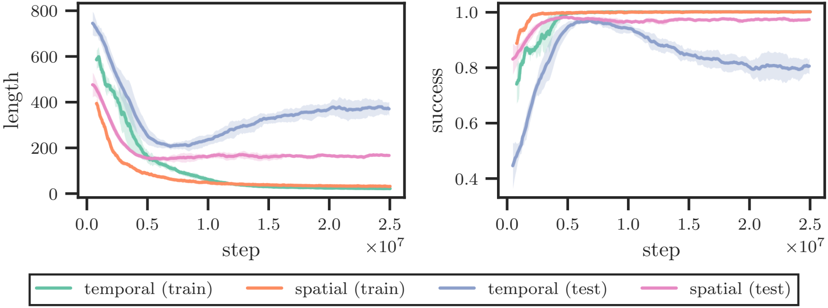
5000 samples



1000 samples



500 samples


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

Conclusion

- Proposed a method for solving visual search with reinforcement learning
- Three environments for evaluating visual search agents
- Compared two neural network architectures with different strengths
- ...

References I

[1] 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.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. Adaptive computation and machine learning series, The MIT Press, second edition ed.

[3] 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.

[4] 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.

References II

[5] 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.

[6] 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.

[7] 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,"

[8] 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].

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

References III

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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,"

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

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

References IV

[18] 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,"