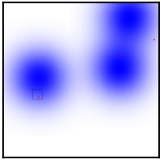
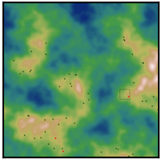
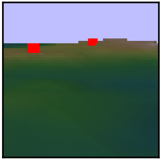
Learning to Search for Targets

with Deep Reinforcement Learning

Oskar Lundin

Linköping University

April 22, 2022

Outline

Problem Description

Autonomous search for a set of targets in an scene with a camera.

- Limited region of scene visible at any given time.
- Camera can be moved to change visible region.
- Locate targets by bringing them into view and indicating that they are visible.
- Should locate all targets while minimizing the number of actions.
- Applications in search and rescue, fire detection, surveillance, etc.

- In a small or random scene with uniformly distributed targets, random or exhaustive search is sufficient.
- Most real-world search tasks are not random, but exhibit structure.
- Cues in the searched scene 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.
- **Can a system learn to search intelligently from a set of samples and generalize to similar search tasks?**

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.
- Real-world tasks have limited number of training samples.

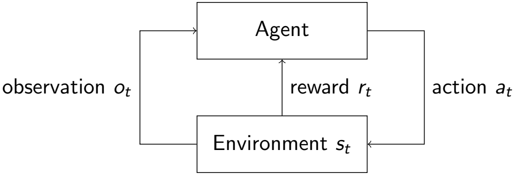
Research Questions

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

Markov Decision Process (MDP)

Framework for modeling decision making in partly random processes.
In our case, *partially observable* MDP [1]:

- *Agent* interacts with *environment* over discrete time steps $t = 0, 1, 2 \ldots, T$.
- Takes *action* a_t in state s_t.
- Perceives (partial) *observation* of state o_t.
- Receives scalar reward r_t that indicates whether action is good or bad.
- New state s_{t+1} depends only on history of interactions.
- Agent usually maintains some internal state depending on history $\rightarrow$ memory.



Reinforcement Learning (RL)

Paradigm for learning from interactions how to achieve a goal.

- Tasks usually formalized as (partially observable) MDPs.
- Policy $\pi(a|s)$ is a mapping from states to actions.
- Find π that maximizes cumulative reward $\mathbb{E}\left[\sum_{k=0}^{T} \gamma^{k-t-1} r_k\right]$.
- Often involves estimating the value $v_\pi(s)$ of a state under policy *pi* (useful for training).

Deep RL: Approximate π (and v_π) with deep neural networks. Has been used to play Atari [2], Go [3], StarCraft II [4], etc.

Search with Reinforcement Learning

- Object localization ([5, 6, 7]).
- Visual navigation (...).
- Todo: add more related work.

Problem Formulation

- Agent searches scene $S \subset \mathbb{R}^d$.
- Scene contains set of targets $\{t_0, \ldots t_n\}$, $t_i \in S$.
- Agent perceives view $V \subset S$.
- Move actions transform view to new subspace.
- Trigger action indicates that a target is in view.
- Select actions that maximize the probability of finding all targets while minimizing cost in time.
- NP complete [8], intractable to solve optimally.

Environments

- Three environments with varying characteristics.
- Search space discretized into 16×16 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.

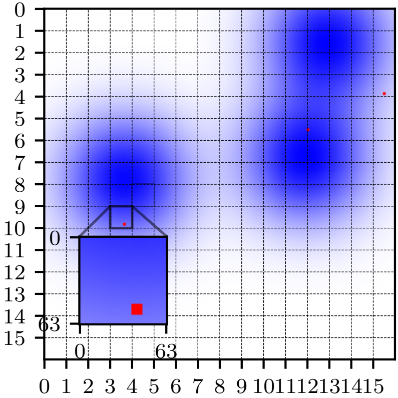
Observation, Action and Reward

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
  - $p_t \in \{0, \ldots, 15\} \times \{0, \ldots, 15\}$ is the position of the camera.
- Takes action $a_t \in \{\texttt{TRIGGER}, \texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}\}$, where
  - `TRIGGER` indicates that a target is in view, and
  - `UP`, `DOWN`, `LEFT`, `RIGHT` move the view in each cardinal direction.
- Receives reward $r_t = h - 0.001$ where $h = |T \cap V|$ is the number of targets in view.
  - Rewarded for finding targets.
  - Constant penalty encourages quick episode completion.

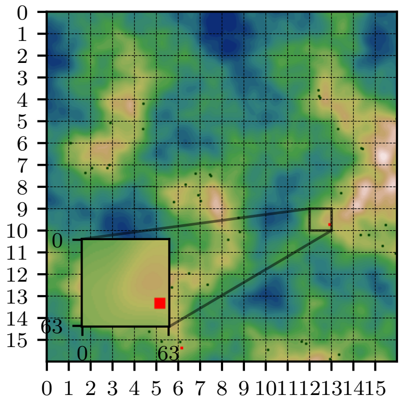
Gaussian Environment

- 2D scene.
- Three gaussian kernels with random center.
- Sum of kernels determine appearance of scene and probability of targets.
- Clear correlation between appearance and desired behavior.



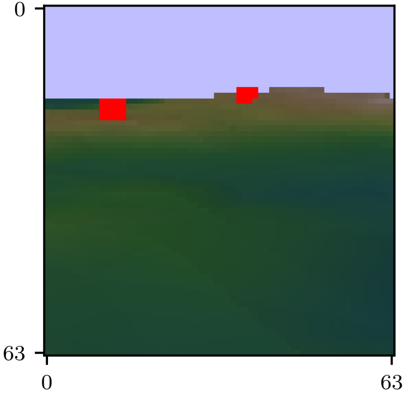
Terrain Environment

- 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.



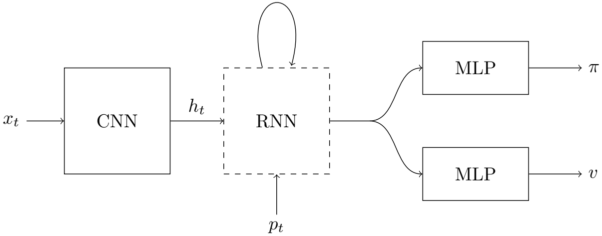
Camera Environment

- 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.



Architecture

- Actor-critic method trained with PPO [9].
- Image x_t passed through CNN.
- Latent image representation h_t and position p_t passed through RNN. Two variants:
  1. LSTM with input $[h_t, p_t]$.
  2. Spatial memory.
- Policy and value heads approximate π and v_π with MLPs.



Recurrent Steps

1. LSTM:
   - Proven to work for POMDPs [10, 11, 12, 13].
   - May struggle with remembering over many time steps.
   - Important for exhaustive search and scene understanding.
2. Spatial memory (inspired by [14]):
   - Structured memory $M_t \in \mathbb{R}^{C\times16\times16}$ 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

- Train for 25M time steps.
- Results reported across 3 runs with different seeds.
- Interval estimates via stratified bootstrap confidence intervals.
- Separate training and test sets.
- Same hyperparameters in all runs.

Experiment I: Search Space and Reward Signal

- Larger search spaces take longer to train:
  - Sparse reward might not be sufficient.
  - Stronger demands on memory (remember searched positions, scene understanding).
- Investigate impact by comparing agents on 8×8, 16×16, 24×24, 32×32 versions of gaussian environment.
- Evaluate two additional reward signals that may speed up training:
  - $r_t' = r_t + e$, where $e = 0.1$ if $a_t \neq \texttt{TRIGGER}$ moves the view to an unexplored region and 0 otherwise.
  - $r_t'' = r_t + d$, where $d = 0.1$ if $a_t \neq \texttt{TRIGGER}$ moves the view towards the nearest target and 0 otherwise.

Experiment II: Search Performance

- Compare to random searcher, exhaustive searcher, human searcher with prior knowledge of scenes.
- Use held out samples as test set.
- Average number of steps on test set.
- SPL metric [15], 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:

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

Experiment III: Generalization with Limited Training Samples

- Limit number of scene samples seen during training to 100, 1000, 10 000, ...
- 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).
- Test on held out scenes from full distribution.

Implementation

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

Preliminary Results

Status:

- Done with implementation.
- Exploratory experiments done, know what seems to work and what does not.
- Spatial memory scales to larger search spaces than LSTM.

Problems:

- Many configurable parts (hyperparameters, environment settings, agent architectures).
- Long training times, difficult to predict hyperparameter interplay, expensive to tune.
- Difficult to design reward signal, magnitudes matter.
- Initial problems with scaling up to large search spaces where intelligent search is more important, hopefully solved now.

Future Steps

1. Collect results across multiple seeds for all experiments.
2. Find opponent.
3. More baselines vs. ablation studies?
4. Discussion and conclusion.
5. Tidy up report.
6. Presentation preparation.

References I

[1] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intelligence*, vol. 101, p. 99–134, May 1998.

[2] 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,” *Nature*, vol. 518, p. 529–533, Feb 2015.

[3] 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,” *Nature*, vol. 529, p. 484–489, Jan 2016.

[4] 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,” *Nature*, vol. 575, p. 350–354, Nov 2019.

References II

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

[6] F. C. Ghesu, B. Georgescu, T. Mansi, D. Neumann, J. Hornegger, and D. Comaniciu, *An Artificial Agent for Anatomical Landmark Detection in Medical Images*, vol. 9902 of *Lecture Notes in Computer Science*, p. 229–237.
Springer International Publishing, 2016.

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

[8] Y. Ye and J. K. Tsotsos, "A complexity-level analysis of the sensor planning task for object search," *Computational Intelligence*, vol. 17, p. 605–620, Nov 2001.
13 citations (Crossref) [2022-02-28].

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347 [cs]*, Aug 2017.
arXiv: 1707.06347.

[10] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *arXiv:1507.06527 [cs]*, Jan 2017.
arXiv: 1507.06527.

References III

[11] 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," *arXiv:1602.01783 [cs]*, Jun 2016.
arXiv: 1602.01783.

[12] 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," *arXiv:1611.03673 [cs]*, Jan 2017.
arXiv: 1611.03673.

[13] S. Gupta, V. Tolani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," *arXiv:1702.03920 [cs]*, Feb 2019.
arXiv: 1702.03920.

[14] E. Parisotto and R. Salakhutdinov, "Neural map: Structured memory for deep reinforcement learning," *arXiv:1702.08360 [cs]*, Feb 2017.
arXiv: 1702.08360.

[15] 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," *arXiv:1807.06757 [cs]*, Jul 2018.
arXiv: 1807.06757.