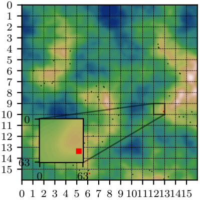
Learning to Search for Targets

with Deep Reinforcement Learning

Oskar Lundin

Linköping University

April 21, 2022



Outline

Problem Description

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

- Limited region of environment observable 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 random environment 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
  - Some are 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.
- Find multiple targets while minimizing path length.
- Search exhaustively while avoiding searching the same region twice.

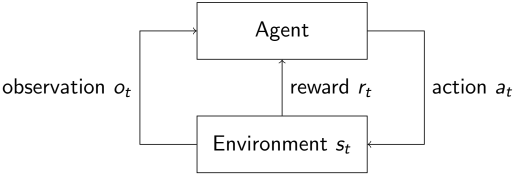
Research Questions

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 searched scene?
3. How does the agent's ability to generalize to unseen in-distribution environments 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:

- *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.
- New state s_{t+1} depends only on history of interactions.
- Agent must maintain some internal state depending on history.



Reinforcement Learning (RL)

Learn from interactions how to achieve a goal.

- Tasks usually formalized as (partially observable) MDPs.
- Policy $\pi(a|s)$ is a mapping from states actions.
- Find π that maximizes cumulative reward $\mathbb{E}\left[\sum_{k=0}^{T} r_k\right]$.
- Often involves estimating

Deep RL: Approximate π (and v_π) with deep neural networks.

Search with Reinforcement Learning

- Object localization ([1, 2, 3]).
- Visual navigation (...).

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 target(s) is in view.
- **Locate all targets while minimizing the number of time steps**.

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.

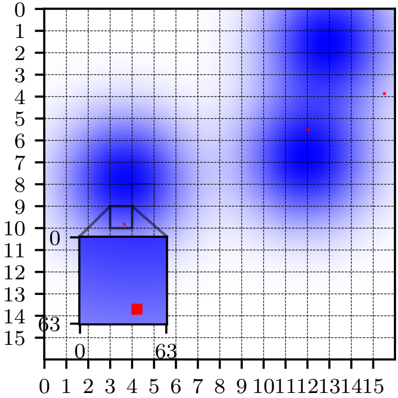
Observation, Action and Reward

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.

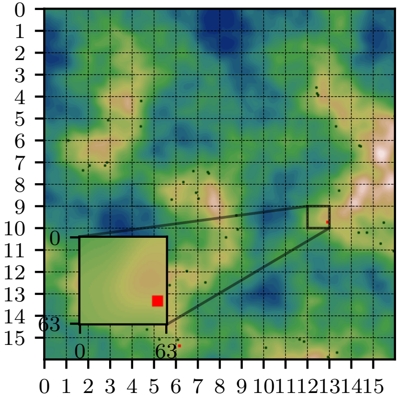
Gaussian Environment

- Two-dimensional 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.



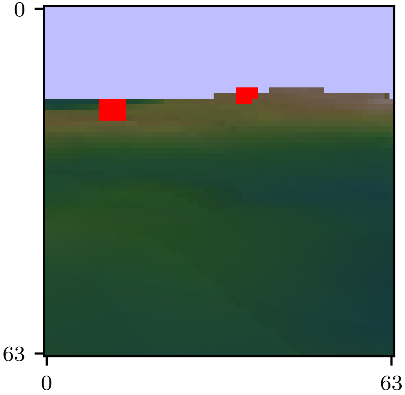
Terrain Environment

- 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.



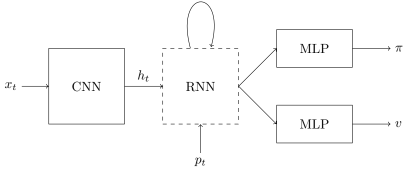
Camera Environment

- 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.



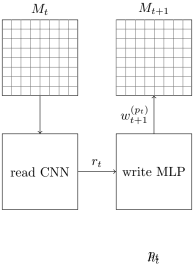
Architecture

- Actor-critic method.
- Trained with proximal policy optimization.
- 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 head approximates π with MLP.
- Value head approximates v with MLP.



Spatial Memory

- LSTM may have difficulties remembering over many time steps and reasoning over spatial relations.
- Specialized memory could be more useful.
- Structured memory with one slot for each camera position.
- Memory read with CNN.
- Written to with previous read and new latent image.



Experiments

- Train for 25M time steps.
- Results reported across 3 runs with different seeds.
- Interval estimates via stratified bootstrap confidence intervals ([4]).
- Separate training and test sets.
- Same hyperparameters in all runs.

Reward signals and search space size:

- 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:

$$r'_t = \begin{cases} 1 & \text{if } a_t \neq \texttt{TRIGGER} \text{ moves view towards nearest target} \\ r_t & \text{otherwise} \end{cases}$$

$$r''_t = \begin{cases} -1 & \text{if } a_t \neq \texttt{TRIGGER} \text{ moves view to visited location} \\ r_t & \text{otherwise} \end{cases}$$

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 [5], 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:

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

Generalization:

- 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:

- Implementation done.
- Test results collected.
- Everything seems to be working.
- Agents can achieve episode lengths that are on par with

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

References I

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

References II

- R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," *arXiv:2108.13264 [cs, stat]*, Jan 2022. arXiv: 2108.13264.
- 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.