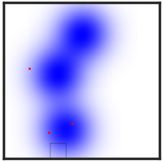
Learning to Search for Targets

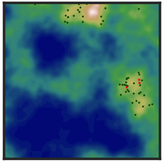
with Deep Reinforcement Learning

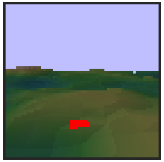
Oskar Lundin

Linköping University

June 12, 2022







Outline

Description

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

- Camera perceives limited region of environment.

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.

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.

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.

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.

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.

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.

Problem Statement

Problem Statement

Problem Statement

- Searched scene $S \subset \mathbb{R}^d$.

Problem Statement

- Searched scene $S \subset \mathbb{R}^d$.
- Perceived view $V \subset S$ in the form of an image.

Problem Statement

- Searched scene $S \subset \mathbb{R}^d$.
- Perceived view $V \subset S$ in the form of an image.
- View can be transformed to new subspace at a cost.

Problem Statement

- Searched scene $S \subset \mathbb{R}^d$.
- Perceived view $V \subset S$ in the form of an image.
- View can be transformed to new subspace at a cost.
- Targets in scene $\{t_0, \dots t_n\}$, $t_i \in S$.

Problem Statement

- Searched scene $S \subset \mathbb{R}^d$.
- Perceived view $V \subset S$ in the form of an image.
- View can be transformed to new subspace at a cost.
- Targets in scene $\{t_0, \ldots t_n\}$, $t_i \in S$.
- Detect when targets are visible, i.e. $V \cap T \neq \varnothing$.

Problem Statement

- Searched scene $S \subset \mathbb{R}^d$.
- Perceived view $V \subset S$ in the form of an image.
- View can be transformed to new subspace at a cost.
- Targets in scene $\{t_0, \ldots t_n\}$, $t_i \in S$.
- Detect when targets are visible, i.e. $V \cap T \neq \varnothing$.
- Goal:

Problem Statement

- Searched scene $S \subset \mathbb{R}^d$.
- Perceived view $V \subset S$ in the form of an image.
- View can be transformed to new subspace at a cost.
- Targets in scene $\{t_0, \dots t_n\}$, $t_i \in S$.
- Detect when targets are visible, i.e. $V \cap T \neq \varnothing$.
- Goal:
    - Maximize probability of finding all targets.

Problem Statement

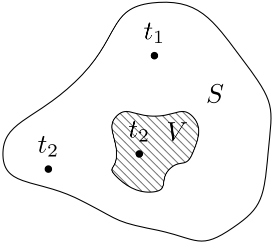
- Searched scene $S \subset \mathbb{R}^d$.
- Perceived view $V \subset S$ in the form of an image.
- View can be transformed to new subspace at a cost.
- Targets in scene $\{t_0, \dots t_n\}$, $t_i \in S$.
- Detect when targets are visible, i.e. $V \cap T \neq \varnothing$.
- Goal:
  - Maximize probability of finding all targets.
  - Minimize cost (time).

Problem Statement

- Searched scene $S \subset \mathbb{R}^d$.
- Perceived view $V \subset S$ in the form of an image.
- View can be transformed to new subspace at a cost.
- Targets in scene $\{t_0, \ldots t_n\}$, $t_i \in S$.
- Detect when targets are visible, i.e. $V \cap T \neq \varnothing$.
- Goal:
  - Maximize probability of finding all targets.
  - Minimize cost (time).
  - NP-complete [1].

Problem Statement

- Searched scene $S \subset \mathbb{R}^d$.
- Perceived view $V \subset S$ in the form of an image.
- View can be transformed to new subspace at a cost.
- Targets in scene $\{t_0, \dots t_n\}$, $t_i \in S$.
- Detect when targets are visible, i.e. $V \cap T \neq \varnothing$.
- Goal:
  - Maximize probability of finding all targets.
  - Minimize cost (time).
  - NP-complete [1].



Motivation

- Applications in search and rescue, surveillance, home assistance, etc.

Motivation

- Applications in search and rescue, surveillance, home assistance, etc.
- Autonomous systems may be faster and less costly than manual ones.

Motivation

- Applications in search and rescue, surveillance, home assistance, etc.
- Autonomous systems may be faster and less costly than manual ones.
- Learning vs. handcrafted systems:

Motivation

- Applications in search and rescue, surveillance, home assistance, etc.
- Autonomous systems may be faster and less costly than manual ones.
- Learning vs. handcrafted systems:
    - May find better solutions (deep RL: Atari [2], Go [3], StarCraft II [4]).

Motivation

- Applications in search and rescue, surveillance, home assistance, etc.
- Autonomous systems may be faster and less costly than manual ones.
- Learning vs. handcrafted systems:
    - May find better solutions (deep RL: Atari [2], Go [3], StarCraft II [4]).
    - Applicable as long as data is available.

Motivation

- Applications in search and rescue, surveillance, home assistance, etc.
- Autonomous systems may be faster and less costly than manual ones.
- Learning vs. handcrafted systems:
  - May find better solutions (deep RL: Atari [2], Go [3], StarCraft II [4]).
  - Applicable as long as data is available.
  - Guarantees and understandability.

Aim

- Utilize structure in environments:

Aim

- Utilize structure in environments:
  - Books are in bookshelves, cars on roads...

Aim

- Utilize structure in environments:
  - Books are in bookshelves, cars on roads...
  - Targets can be spread out/close together...

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.

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.

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 scenarios.

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 scenarios.
- Remember features of explored environment to:

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 scenarios.
- Remember features of explored environment to:
  - Avoid searching regions twice.

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 scenarios.
- 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?

Research Questions

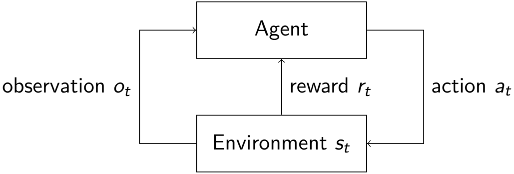
1. How can an agent that learns to intelligently search for targets be implemented with deep reinforcement learning?
2. How does the learning agent compare to random, greedy, exhaustive and human searchers?

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?
3. How well does the learning agent generalize from a limited number of training samples to unseen in-distribution search scenarios?

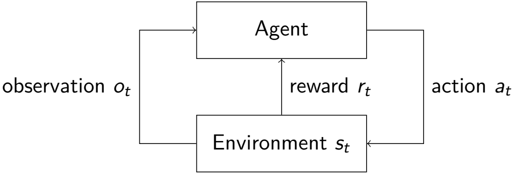
Reinforcement Learning I

- Learn from interaction how to achieve a goal.



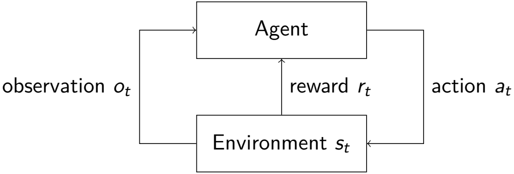
Reinforcement Learning I

- Learn from interaction how to achieve a goal.
- Partially Observable Markov Decision Process [5]:



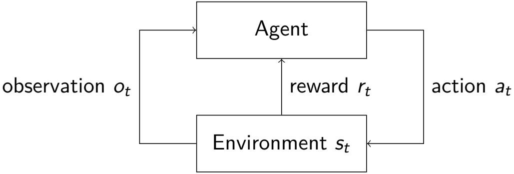
Reinforcement Learning I

- Learn from interaction how to achieve a goal.
- Partially Observable Markov Decision Process [5]:
  - *Agent* interacts with *environment* over discrete time steps $t = 0, 1, 2 \ldots, T$.



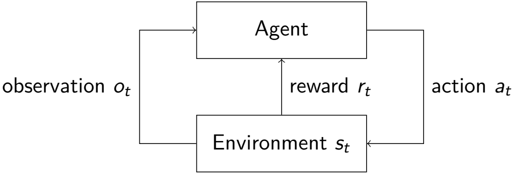
Reinforcement Learning I

- Learn from interaction how to achieve a goal.
- Partially Observable Markov Decision Process [5]:
  - *Agent* interacts with *environment* over discrete time steps $t = 0, 1, 2 \ldots, T$.
  - New state s_{t+1} depends on history $a_0, o_1, r_1, \ldots, a_{t-1}, o_t, r_t$.



Reinforcement Learning I

- Learn from interaction how to achieve a goal.
- Partially Observable Markov Decision Process [5]:
    - *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 $\pi(a|s)$ defines agent's behavior.

Reinforcement Learning II

- Policy $\pi(a|s)$ defines agent's behavior.
- Find policy that maximizes expected future reward $\mathbb{E}\left[\sum_{k=0}^{T} \gamma^{k-t-1} r_k\right]$.

Reinforcement Learning II

- Policy $\pi(a|s)$ defines agent's behavior.
- Find policy that maximizes expected future reward $\mathbb{E}\left[\sum_{k=0}^{T} \gamma^{k-t-1} r_k\right]$.
- There are several different algorithms.

Reinforcement Learning II

- Policy $\pi(a|s)$ defines agent's behavior.
- Find policy that maximizes expected future reward $\mathbb{E}\left[\sum_{k=0}^{T} \gamma^{k-t-1} r_k\right]$.
- There are several different algorithms.
- Reward signal is often a design parameter.

Reinforcement Learning II

- Policy $\pi(a|s)$ defines agent's behavior.
- Find policy that maximizes expected future reward $\mathbb{E}\left[\sum_{k=0}^{T} \gamma^{k-t-1} r_k\right]$.
- There are several different algorithms.
- Reward signal is often a design parameter.
- Deep reinforcement learning: approximate π with deep neural networks.

Related Work

- Visual attention (determining what to pay attention to in a visual environment):

Related Work

- Visual attention (determining what to pay attention to in a visual environment):
  - Sequential focus points for foveated vision. [6]

Related Work

- Visual attention (determining what to pay attention to in a visual environment):
  - Sequential focus points for foveated vision. [6]
- Visual navigation (searching for a goal location in a visual environment):

Related Work

- Visual attention (determining what to pay attention to in a visual environment):
  - Sequential focus points for foveated vision. [6]
- Visual navigation (searching for a goal location in a visual environment):
  - Solve random mazes [7].

Related Work

- Visual attention (determining what to pay attention to in a visual environment):
  - Sequential focus points for foveated vision. [6]
- Visual navigation (searching for a goal location in a visual environment):
  - Solve random mazes [7].
  - Find target object in indoor scenes [8].

Related Work

- Visual attention (determining what to pay attention to in a visual environment):
  - Sequential focus points for foveated vision. [6]
- Visual navigation (searching for a goal location in a visual environment):
  - Solve random mazes [7].
  - Find target object in indoor scenes [8].
- Object localization (searching for objects in an image):

Related Work

- Visual attention (determining what to pay attention to in a visual environment):
  - Sequential focus points for foveated vision. [6]
- Visual navigation (searching for a goal location in a visual environment):
  - Solve random mazes [7].
  - Find target object in indoor scenes [8].
- Object localization (searching for objects in an image):
  - Region proposals for object localization [9].

Related Work

- Visual attention (determining what to pay attention to in a visual environment):
  - Sequential focus points for foveated vision. [6]
- Visual navigation (searching for a goal location in a visual environment):
  - Solve random mazes [7].
  - Find target object in indoor scenes [8].
- Object localization (searching for objects in an image):
  - Region proposals for object localization [9].
  - Anatomical landmark detection in medical images [10].

Environments

- Three simulated environments.

Environments

- Three simulated environments.
- Structure can be utilized to find targets quicker.

Environments

- Three simulated environments.
- Structure can be utilized to find targets quicker.
- Procedurally generated, conditioned on seed.

Environments

- Three simulated environments.
- Structure can be utilized to find targets quicker.
- Procedurally generated, conditioned on seed.
- Find three targets in less than 1 000 steps.

Environments

- Three simulated environments.
- Structure can be utilized to find targets quicker.
- Procedurally generated, conditioned on seed.
- Find three targets in less than 1 000 steps.
- New seed after each finished search.

Observation, Action and Reward

- Observations $o_t = \langle x_t, p_t \rangle$, where

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,

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.

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,
  - $p_t \in \{0, \ldots, H\} \times \{0, \ldots, W\}$ is the camera position.
- Actions $a_t \in \{\texttt{INDICATE}, \texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}\}$, where

Observation, Action and Reward

- Observations $o_t = \langle x_t, p_t \rangle$, where
  - $x_t \in \mathbb{R}^{3\times64\times64}$ is an RGB image,
  - $p_t \in \{0, \ldots, H\} \times \{0, \ldots, W\}$ is the camera position.
- Actions $a_t \in \{\texttt{INDICATE}, \texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}\}$, where
  - `INDICATE` identifies targets, and

Observation, Action and Reward

- Observations $o_t = \langle x_t, p_t \rangle$, where
  - $x_t \in \mathbb{R}^{3\times64\times64}$ is an RGB image,
  - $p_t \in \{0, \dots, H\} \times \{0, \dots, W\}$ is the camera position.
- Actions $a_t \in \{\mathtt{INDICATE}, \mathtt{UP}, \mathtt{DOWN}, \mathtt{LEFT}, \mathtt{RIGHT}\}$, where
  - `INDICATE` identifies targets, and
  - `UP`, `DOWN`, `LEFT`, `RIGHT` move camera.

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,
  - $p_t \in \{0, \ldots, H\} \times \{0, \ldots, W\}$ 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

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.

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.

Observation, Action and Reward

- Observations $o_t = \langle x_t, p_t \rangle$, where
  - $x_t \in \mathbb{R}^{3\times64\times64}$ 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

Environment I: Gaussian

Environment I: Gaussian

- Three gaussian distributions with random center.

Environment I: Gaussian

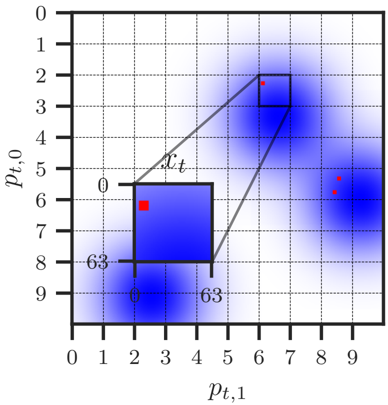
- Three gaussian distributions with random center.
- Normalized sum give blue color intensity and target probability.

Environment I: Gaussian

- Three gaussian distributions with random center.
- Normalized sum give blue color intensity and target probability.
- Agent should prioritize blue regions.

Environment I: Gaussian

- Three gaussian distributions with random center.
- Normalized sum give blue color intensity and target probability.
- Agent should prioritize blue regions.



Environment II: Terrain

Environment II: Terrain

Environment II: Terrain

- Terrain seen from above (e.g. UAV).

Environment II: Terrain

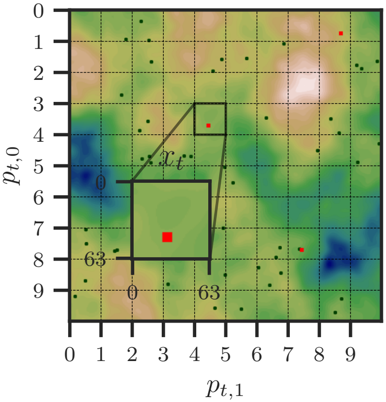
- Terrain seen from above (e.g. UAV).
- Targets between ocean and mountains.

Environment II: Terrain

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

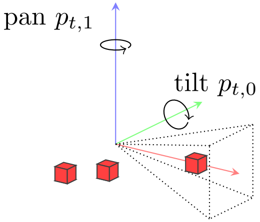
Environment II: Terrain

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



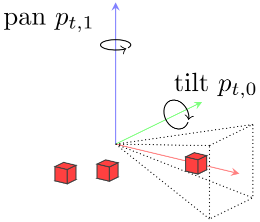
Environment III: Camera

Environment III: Camera



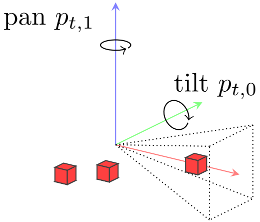
Environment III: Camera

- Terrain seen from perspective projection camera.



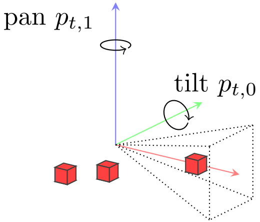
Environment III: Camera

- Terrain seen from perspective projection camera.
- Moving actions control pan and tilt.



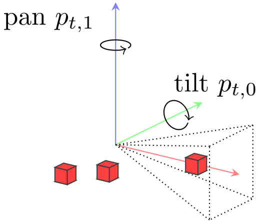
Environment III: Camera

- Terrain seen from perspective projection camera.
- Moving actions control pan and tilt.
  - 20 pan angle steps.



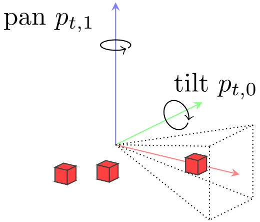
Environment III: Camera

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



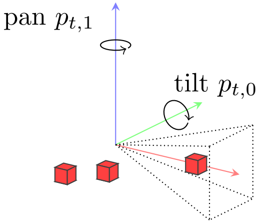
Environment III: Camera

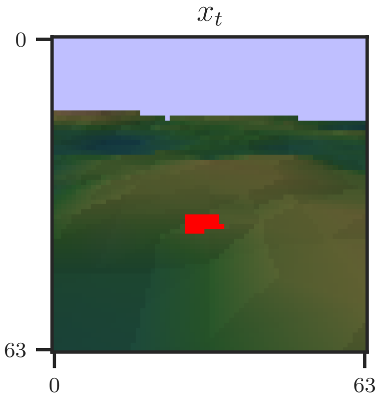
- Terrain seen from perspective projection camera.
- Moving actions control pan and tilt.
  - 20 pan angle steps.
  - 10 tilt angle steps.
- Variance in target appearance.



Environment III: Camera

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





Approach

- Function approximation with deep neural networks.

Approach

- Function approximation with deep neural networks.
  - Policy $\pi(a|s, \theta)$.

Approach

- Function approximation with deep neural networks.
  - Policy $\pi(a|s, \theta)$.
  - Value $v_\pi(s, \theta)$ (predicts future reward).

Approach

- Function approximation with deep neural networks.
  - Policy $\pi(a|s, \theta)$.
  - Value $v_\pi(s, \theta)$ (predicts future reward).
- Training procedure:

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.

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 $\mathcal{L}(\theta)$.

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.

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 until π achieves high reward.

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 until π achieves high reward.
- Reinforcement learning algorithm:

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 until π achieves high reward.
- Reinforcement learning algorithm:
  - Proximal policy optimization [11].

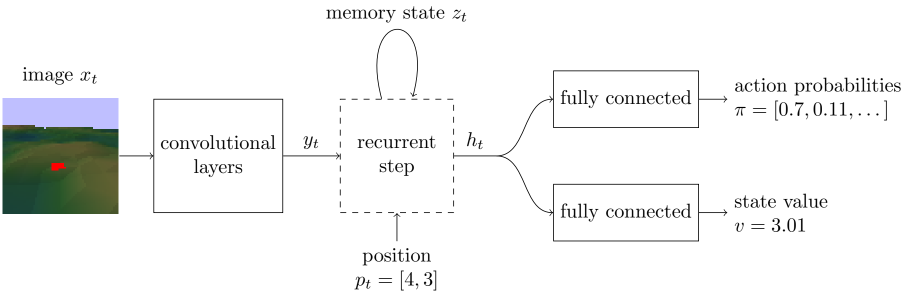
Approach

- 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 until π achieves high reward.
- Reinforcement learning algorithm:
  - Proximal policy optimization [11].
  - Relatively new algorithm.

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 until π achieves high reward.
- Reinforcement learning algorithm:
  - Proximal policy optimization [11].
  - Relatively new algorithm.
  - Stable performance with little tuning [12].

Architecture



Memory

- *Agent should remember visual features and associate them with their spatial location.*

Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:

Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:
  1. Temporal memory (long short-term memory [13]):

Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:
  1. Temporal memory (long short-term memory [13]):
     - Previously applied successfully to tasks where memory is required [14, 15].

Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:
  1. Temporal memory (long short-term memory [13]):
     - Previously applied successfully to tasks where memory is required [14, 15].
     - How long sequences can be remembered?

Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:
  1. Temporal memory (long short-term memory [13]):
     - Previously applied successfully to tasks where memory is required [14, 15].
     - How long sequences can be remembered?
  2. Spatial memory (inspired by [16] and [17]):

Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:
  1. Temporal memory (long short-term memory [13]):
     - Previously applied successfully to tasks where memory is required [14, 15].
     - How long sequences can be remembered?
  2. Spatial memory (inspired by [16] and [17]):
     - Feature map with one slot per camera position.

Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:
  1. Temporal memory (long short-term memory [13]):
     - Previously applied successfully to tasks where memory is required [14, 15].
     - How long sequences can be remembered?
  2. Spatial memory (inspired by [16] and [17]):
     - Feature map with one slot per camera position.
     - Indexed with current position.

Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:
  1. Temporal memory (long short-term memory [13]):
     - Previously applied successfully to tasks where memory is required [14, 15].
     - How long sequences can be remembered?
  2. Spatial memory (inspired by [16] and [17]):
     - Feature map with one slot per camera position.
     - Indexed with current position.
     - Stores image representation at each slot.

Memory

- *Agent should remember visual features and associate them with their spatial location.*
- Two memory variants:
  1. Temporal memory (long short-term memory [13]):
     - Previously applied successfully to tasks where memory is required [14, 15].
     - How long sequences can be remembered?
  2. Spatial memory (inspired by [16] and [17]):
     - 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.

Training

- Train for 25M time steps.
- Results reported across 3 training runs.

Training

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

Implementation

- OpenAI Gym environment interface.

Implementation

- OpenAI Gym environment interface.
- Custom proximal policy optimization implementation.

Implementation

- OpenAI Gym environment interface.
- Custom proximal policy optimization implementation.
- PyTorch for models and automatic differentiation.

Implementation

- OpenAI Gym environment interface.
- Custom proximal policy optimization implementation.
- PyTorch for models and automatic differentiation.
- Intel Core i9-10900X CPU.

Implementation

- OpenAI Gym environment interface.
- Custom proximal policy optimization 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).

Experiment I: Search Performance

- Compare to simple reference behaviors (baselines).
- Fixed test set from each environment.

Experiment I: Search Performance

- Compare to simple reference behaviors (baselines).
- Fixed test set from each environment.
- Metrics:

Experiment I: Search Performance

- Compare to simple reference behaviors (baselines).
- Fixed test set from each environment.
- Metrics:
  1. Average search path length.

Experiment I: Search Performance

- Compare to simple reference behaviors (baselines).
- Fixed test set from each environment.
- Metrics:
  1. Average search path length.
  2. Average success rate.

Experiment I: Search Performance

- Compare to simple reference behaviors (baselines).
- Fixed test set from each environment.
- Metrics:
    1. Average search path length.
    2. Average success rate.
    3. Success weighted by inverse path length (SPL) [18].
       With N test samples, S_i as a binary success indicator, p_i as the taken search path length l_i is the shortest search path length:

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

Baselines

Random: randomly samples actions.

Baselines

Random: randomly samples actions.

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

Baselines

Random: randomly samples actions.

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

Exhaustive: exhaustively covers search space with minimal revisits.

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.

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

- Real-world search tasks usually have large search spaces.

Experiment II: Scaling to Larger Search Spaces

- Real-world search tasks usually have large search spaces.
- Stronger demands on memory:

Experiment II: Scaling to Larger Search Spaces

- Real-world search tasks usually have large search spaces.
- Stronger demands on memory:
  - Remember visited positions.

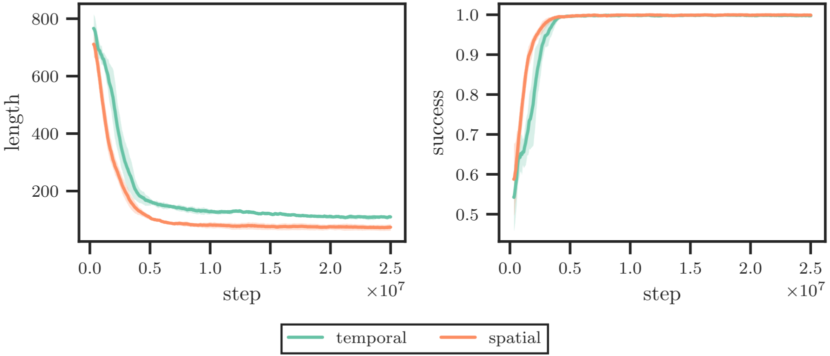
Experiment II: Scaling to Larger Search Spaces

- Real-world search tasks usually have large search spaces.
- Stronger demands on memory:
  - Remember visited positions.
  - Remember appearance of environment.

Experiment II: Scaling to Larger Search Spaces

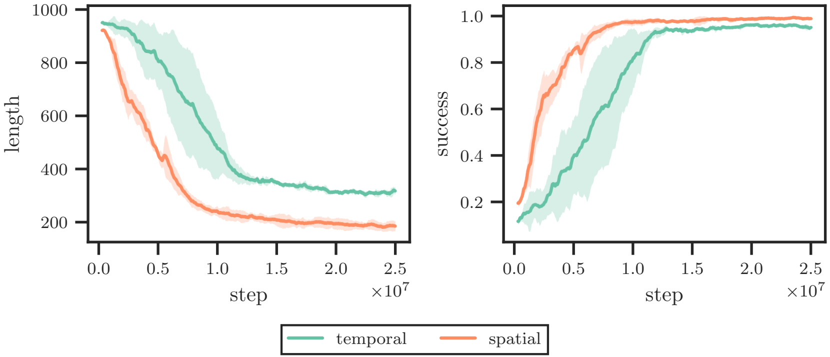
- Real-world search tasks usually have large search spaces.
- Stronger demands on memory:
  - Remember visited positions.
  - Remember appearance of environment.
- Compare memories on 10×10, 15×15, and 20×20 versions of gaussian environment.

10×10

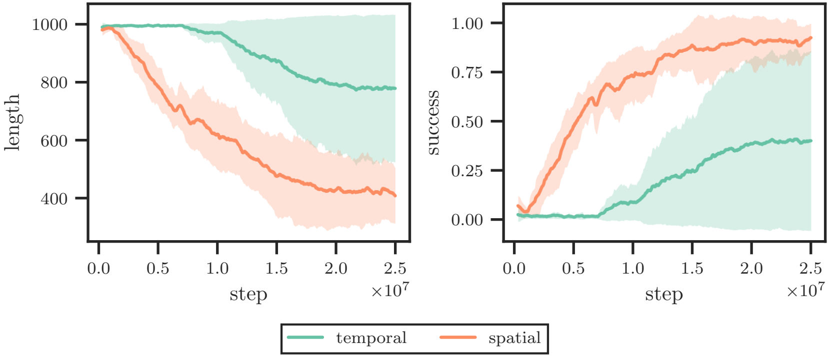


length
800
600
400
200
0.0
0.5
1.0
1.5
2.0
2.5
step
×10^7
success
1.0
0.9
0.8
0.7
0.6
0.5
temporal
spatial

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
0.0
0.5
1.0
1.5
2.0
2.5
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
temporal
spatial

Experiment III: Generalization From Limited Samples

- Real-world tasks usually have limited training samples.

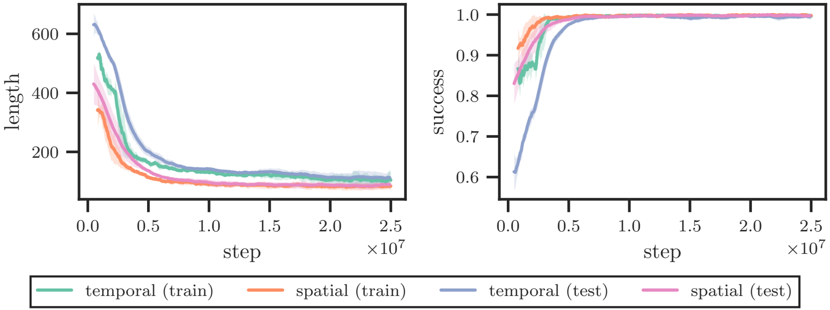
Experiment III: Generalization From Limited Samples

- Real-world tasks usually have limited training samples.
- Train on 500, 1 000, 5 000 and 10 000 samples of terrain environment.

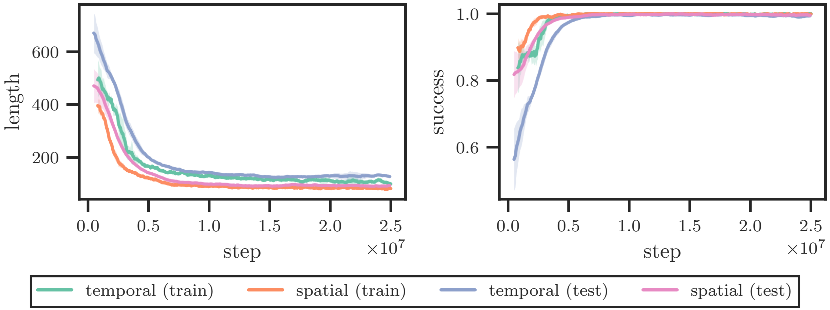
Experiment III: Generalization From Limited Samples

- 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.

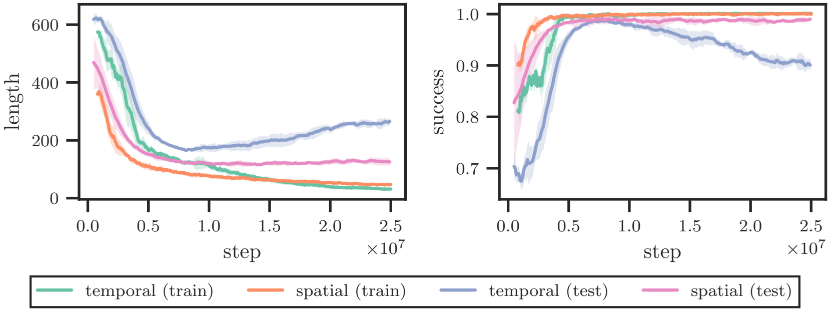
10000 samples



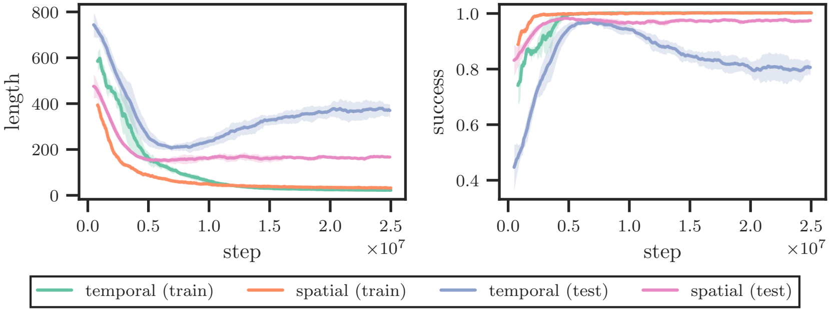
5000 samples



1000 samples



500 samples



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

Conclusion

- Architecture:

Conclusion

- Architecture:
  - Spatial memory: architecture scales to larger search spaces and generalizes better.

Conclusion

- Architecture:
  - Spatial memory: architecture scales to larger search spaces and generalizes better.
  - Temporal memory: sufficient (and better) for smaller search spaces.

Conclusion

- Architecture:
  - Spatial memory: architecture scales to larger search spaces and generalizes better.
  - Temporal memory: sufficient (and better) for smaller search spaces.
- Approach:

Conclusion

- Architecture:
  - Spatial memory: architecture scales to larger search spaces and generalizes better.
  - Temporal memory: sufficient (and better) for smaller search spaces.
- Approach:
  - Search performance: better than simple baselines, comparable to human, worse than handcrafted.

Conclusion

- Architecture:
  - Spatial memory: architecture scales to larger search spaces and generalizes better.
  - Temporal memory: sufficient (and better) for smaller search spaces.
- Approach:
  - Search performance: better than simple baselines, comparable to human, worse than handcrafted.
  - Sample efficiency: relatively many samples needed even for simple environments.

Future Work

- Improvements to approach.

Future Work

- Improvements to approach.
  - Neural network architecture.

Future Work

- Improvements to approach.
  - Neural network architecture.
  - Reinforcement learning algorithm.

Future Work

- Improvements to approach.
  - Neural network architecture.
  - Reinforcement learning algorithm.
  - Reward signal design.

Future Work

- Improvements to approach.
  - Neural network architecture.
  - Reinforcement learning algorithm.
  - Reward signal design.
- Evaluate on realistic search scenarios.

Future Work

- Improvements to approach.
  - Neural network architecture.
  - Reinforcement learning algorithm.
  - Reward signal design.
- Evaluate on realistic search scenarios.
  - Does the approach scale?

Future Work

- Improvements to approach.
  - Neural network architecture.
  - Reinforcement learning algorithm.
  - Reward signal design.
- Evaluate on realistic search scenarios.
  - Does the approach scale?
  - Difficult detection problems.

Future Work

- Improvements to approach.
  - Neural network architecture.
  - Reinforcement learning algorithm.
  - Reward signal design.
- Evaluate on realistic search scenarios.
  - Does the approach scale?
  - Difficult detection problems.
  - Noise and higher variance.

References I

[1] 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.

References II

[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," vol. 575, no. 7782, pp. 350–354.
Number: 7782 Publisher: Nature Publishing Group.

[5] 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.

[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].

References III

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

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

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

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

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

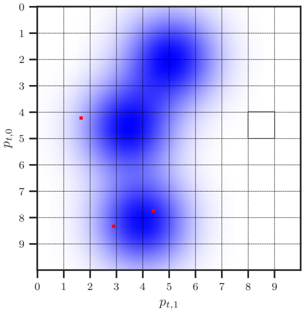
[17] S. Gupta, V. Tolani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation,"

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,"

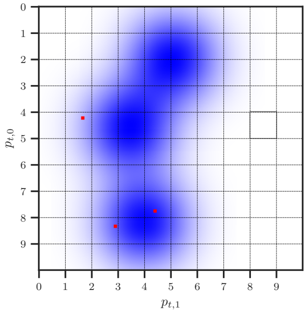
Search Paths I

Search Paths I

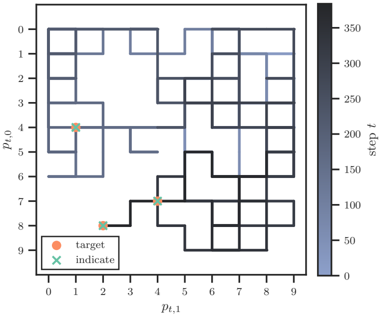


Environment sample

Search Paths I



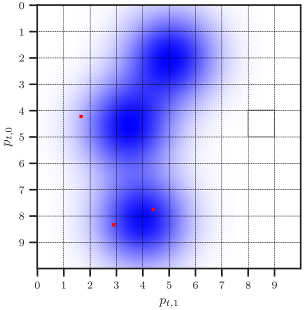
Environment sample



Random baseline

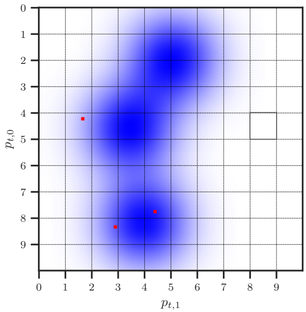
Search Paths II

Search Paths II

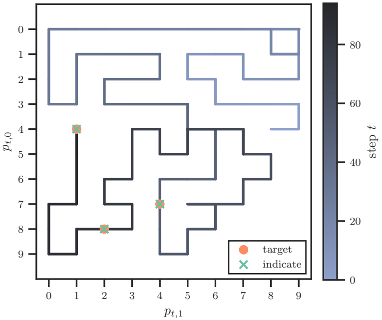


Environment sample

Search Paths II



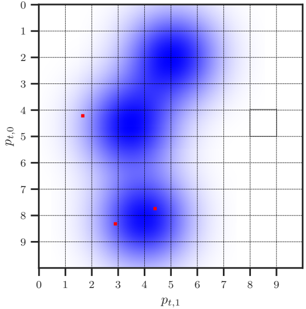
Environment sample



Greedy baseline

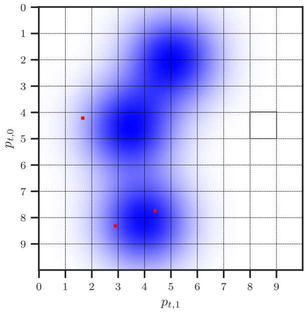
Search Paths III

Search Paths III

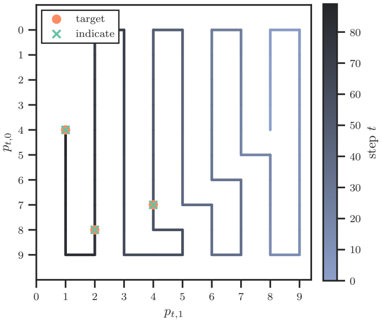


Environment sample

Search Paths III



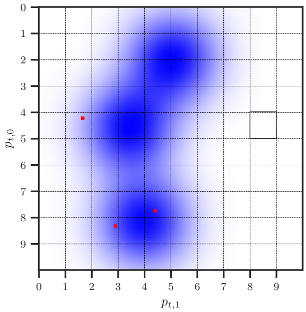
Environment sample



Exhaustive baseline

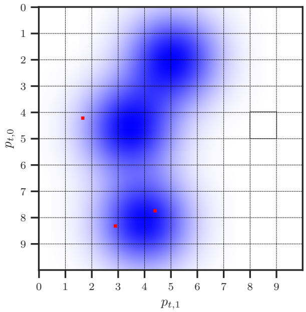
Search Paths IV

Search Paths IV

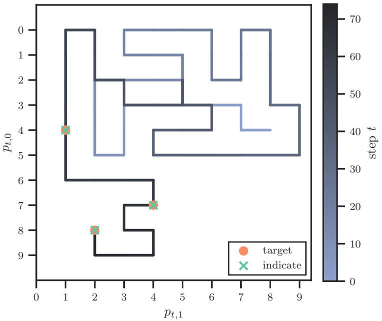


Environment sample

Search Paths IV



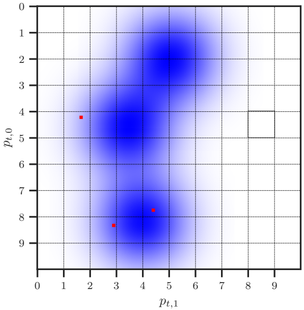
Environment sample



Handcrafted baseline

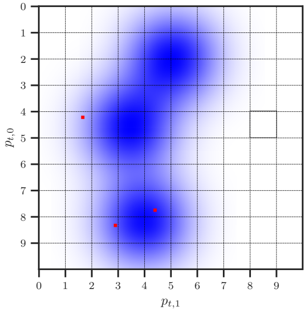
Search Paths V

Search Paths V

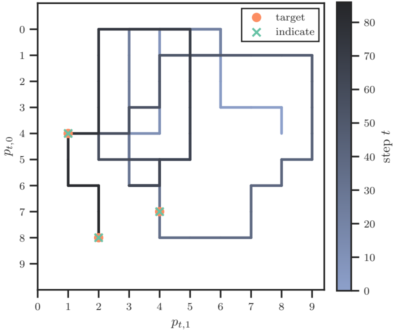


Environment sample

Search Paths V



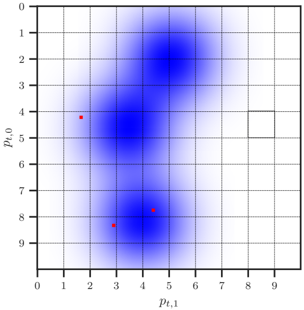
Environment sample



Temporal memory

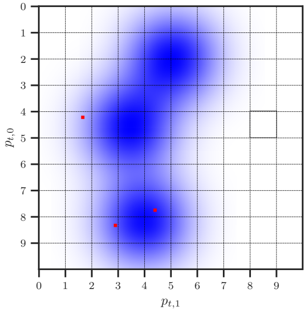
Search Paths VI

Search Paths VI

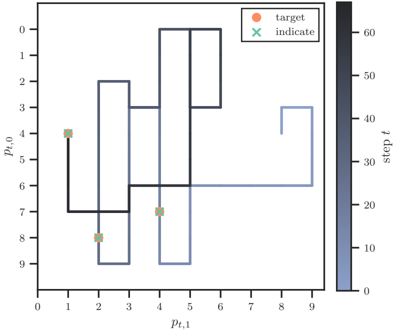


Environment sample

Search Paths VI

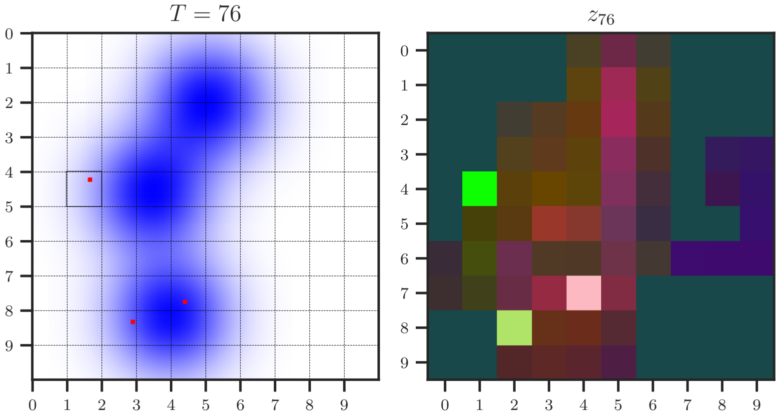


Environment sample



Spatial memory

Memory Viualization



PCA decomposition of spatial memory after episode.