Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Datateknik
2017 | LIU-IDA/LITH-EX-A--2017/001--SE

Learning to Search for Targets

– A Deep Reinforcement Learning Approach for Scanning Large Environments

Inlärd sökning efter mål

Oskar Lundin

Supervisor : Sourabh Balgi
Examiner : Jose M. Peña

External supervisor : Fredrik Bissmarck

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

The abstract resides in file `Abstract.tex`. Here you should write a short summary of your work.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in massa suscipit, congue massa in, pharetra lacus. Donec nec felis tempor, suscipit metus molestie, consectetur orci. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Curabitur fermentum, augue non ullamcorper tempus, ex urna suscipit lorem, eu consectetur ligula orci quis ex. Phasellus imperdiet dolor at luctus tempor. Curabitur nisi enim, porta ut gravida nec, feugiat fermentum purus. Donec hendrerit justo metus. In ultrices malesuada erat id scelerisque. Sed sapien nisi, feugiat in ligula vitae, condimentum accumsan nisi. Nunc sit amet est leo. Quisque hendrerit, libero ut viverra aliquet, neque mi vestibulum mauris, a tincidunt nulla lacus vitae nunc. Cras eros ex, tincidunt ac porta et, vulputate ut lectus. Curabitur ultricies faucibus turpis, ac placerat sem sollicitudin at. Ut libero odio, eleifend in urna non, varius imperdiet diam. Aenean lacinia dapibus mauris. Sed posuere imperdiet ipsum a fermentum.

Nulla lobortis enim ac magna rhoncus, nec condimentum erat aliquam. Nullam laoreet interdum lacus, ac rutrum eros dictum vel. Cras lobortis egestas lectus, id varius turpis rhoncus et. Nam vitae auctor ligula, et fermentum turpis. Morbi neque tellus, dignissim a cursus sed, tempus eu sapien. Morbi volutpat convallis mauris, a euismod dui egestas sit amet. Nullam a volutpat mauris. Fusce sed ipsum lectus. In feugiat, velit eu fermentum efficitur, mi ex eleifend ante, eget scelerisque sem turpis nec augue.

Vestibulum posuere nibh ut iaculis semper. Ut diam justo, interdum quis felis ac, posuere fermentum ex. Fusce tincidunt vel nunc non semper. Sed ultrices suscipit dui, vel lacinia lorem euismod quis. Etiam pellentesque vitae sem eu bibendum. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Pellentesque scelerisque congue ullamcorper. Sed vehicula sodales velit a scelerisque. Pellentesque dignissim lectus ipsum, quis consectetur tellus rhoncus a.

Nunc placerat ut lectus vel ornare. Sed nec dictum enim. Donec imperdiet, ipsum ut facilisis blandit, lacus nisi maximus ex, sed semper nisl metus eget leo. Nunc efficitur risus ac risus placerat, vel ullamcorper felis interdum. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Duis vitae felis vel nibh sodales fringilla. Donec semper eleifend sem quis ornare. Proin et leo ut dolor consectetur vehicula. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nunc dignissim interdum orci, sit amet pretium nibh consectetur sagittis. Aenean a eros id risus aliquam placerat nec ut lectus. Curabitur at quam in nisi sodales imperdiet in at erat. Praesent euismod pulvinar imperdiet. Nam auctor mattis nisi in efficitur. Quisque non cursus ipsum, consequat vehicula justo. Fusce varius metus et nulla rutrum scelerisque. Praesent molestie elementum nulla a consequat. In at facilisis nisi, convallis molestie sapien. Cras id ullamcorper purus. Sed at lectus sit amet dolor finibus suscipit vel et purus. Sed odio ipsum, dictum vel justo sit amet, interdum dictum justo. Quisque euismod quam magna, at dignissim eros varius in. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Acknowledgments

Acknowledgments.tex

Contents

List of Figures

Notation

x	variable
X	random variable
$\mathbb{X}$	set

1 Introduction

In this thesis project, the problem of searching for targets in unknown but familiar environments is addressed. This chapter presents the motivation behind the project, the research questions that are addressed, and the delimitations.

1.1 Motivation

The ability to visually search for targets in an environment is crucial to many parts of our daily lives. We are constantly looking for things, be it the right book in the bookshelf, a certain keyword in an article or blueberries in the forest. In many cases, it is important that this search is efficient and fast. Animals need to quickly identify predators, and drivers need to be able to search for pedestrians crossing the road they are driving on.

While searching for targets is often seemingly effortless to humans, it is a complex process. How humans and animals search for things has been extensively studied in neuroscience and neurobiology [11, 30, 29]. Applications from automated search and rescue to helping robots mean that it is of great interest to automate visual search. In the computer vision field, there has been several attempts to mimic the way humans search in machines []. Most attempts focus on fully observable scenes where the target is in view and the task is to localize it (object localization). However, in many real-world visual search scenarios the field-of-view is limited. This means that the search process is split into two steps: directing the field of view (covert attention), and locating targets within the view (overt attention). Much work has been focused on latter, locating targets within the field of view [].

When only a fraction of the environment is visible, where to move the field of view becomes an important decision. The characteristics of the searched environment can often be used to find targets quicker. For example, if one is foraging for blueberries it makes sense to search the ground rather than the trees. Similarly, if one is searching a satellite image for boats it is reasonable to focus on ocean shores. If you see a railroad track or the wake of a boat you can usually follow it to find a vehicle. The exact characteristics of the environment need not be constant - forests with blueberries can vary greatly in appearance and boats can be found in all of the seven seas. In many cases, the environment is familiar in that it has characteristics that are similar to previously seen environments. Humans are able to generalize in such cases.

Manually creating search algorithms for such tasks is problematic. The appearance and distribution of targets in an environment varies greatly, and may be subtle. The visual richness of the environment itself is another problem. How can you identify useful hints from the environment to guide covert attention? Manually engineering such a platform seems infeasible. If one could instead learn the underlying from a limited set of sample environments and generalize to unseen similar environments this problem would be circumvented.

Finally, one wants to avoid looking at the same region twice. The need for keeping track of explored areas necessites some kind of memory.

1.2 Aim

This work tries to address these issues, focusing on strategic scans of larger environments where the field of view is small relative to the environment. This is a problem that has been less studied in the literature than visual search in smaller environments. There are other factors that become increasingly important. The field-of-view of the observer is often limited, and she has to move it efficiently to find the target.

The aim of this thesis is to implement and evaluate an autonomous agent that intelligently searches its environment for targets. The agent should learn common characteristics of environments and utilize this knowledge to search for targets in new environments more effectively. Furthermore, the agent should be able to generalize to unseen environments drawn from the same distribution as the ones it has seen previously.

A specific instance of the visual search problem is considered, where the environment is searched by a pan-tilt camera fixed in place. The camera has a limited view of the environent. Automating this task is of interest for multiple reasons. Manually controlling a camera may be costly, and the performance of a human operator may be suboptimal. Crucial to the problem is generalization.

1.3 Research questions

This thesis will address the following questions:

1. How can a learning agent that does efficient visual search in familiar environments be implemented?
2. How does the learning agent compare to random walk, exhaustive search, frontier-based search, a human searcher, and standard RL methods?
3. How well does the learning agent generalize to unseen but familiar environments?

1.4 Delimitations

This thesis will be focused on the behavioral aspects of the presented problem. We do not focus on difficult detection problems, but rather efficient actions. To train and test agents, a simplified environment will be used. This will test the desired characteristics of the agent as presented above, but will not simulate realistic environments. For simplicity, we assume that the environment is static. We also focus on the search process and not the detection, and therefore targets will be easy to detect once visible.

2 Theory

This chapter introduces relevant theory and related work. Section 2.3.2 gives some background on active vision. Section 2.3.3 describes the problem of visual searching for targets.

2.1 Artificial Neural Networks

An artificial neural network (ANN) is a type of universal function approximator. ANNs are based on

2.1.1 Feed-forward Neural Network

A feed-forward neural network, also called multi-layer perceptron (MLP) defines a mapping $y = f(x;\theta)$. The value of the parameter θ is learned

[13]

2.1.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are designed specifically for processing data with a known grid-like topology, such as images. As the name implies, CNNs employ the convolution operator

$$s(t)$$

[13]

2.1.3 Recurrent Neural Network

Recurrent neural networks (RNNs) are networks designed to process sequential data. By sharing parameters across different parts of a network, it is possible to generalize to

A commonly used RNN architecture is long short-term memory (LSTM) [19].

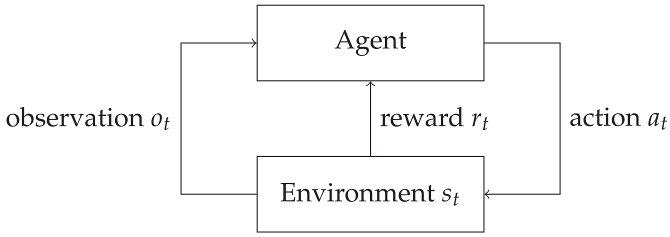

Figure 2.1: Partially observable Markov decision process.

2.2 Reinforcement Learning

Reinforcement learning (RL) [28] is a subfield of machine learning concerned with learning from interaction how to achieve a goal. This section introduces the fundamental concepts of RL.

2.2.1 Partially Observable Markov Decision Processes

The problem of learning from interaction to achieve some goal is often framed as a (finite) Markov decision process (MDP). A learning *agent* interacts continually with its *environment*. The agent takes the *state* of the environment as input, and select an *action* to take. This action updates the state of the environment and gives the agent a scalar *reward*. It is assumed that the next state and reward depend only on the previous state and the action taken. This is referred to as the *Markov* property. [20]

In an MDP, the agent can perceive the state of the environment with full certainty. For many problems, including the one we consider here, this is not the case. The agent can only perceive a partial representation of the environment's state. Such a process is referred to as a partially observable Markov decision process (POMDP). A POMDP is formally defined as a 7-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma \rangle$, where

- $\mathcal{S}$ is a finite set of states,
- $\mathcal{A}$ is a finite set of actions,
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ is a state-transition function,
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function,
- Ω is a finite set of observations,
- $\mathcal{O} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\Omega)$ is an observation function, and
- $\gamma \in [0, 1]$ is a discount factor.

Assume that the environment is in state $s \in \mathcal{S}$, and the agent selects action $a \in \mathcal{A}$. Then, $T(s, a, s')$ is the probability of ending in state s' and $R(s, a)$ is the expected reward gained by the agent. The agent also receives an observation $o \in \Omega$ with probability $\mathcal{O}(s', a, o)$. The agent and environment interact over a sequence of discrete time steps $t = 0, 1, \ldots, T$, giving rise to an *episode* of length T. The goal of the agent is to maximize the expected *discounted return* $\mathbb{E}[\Sigma_{t=0}^{\infty} \gamma^t r_t]$. [20]

Figure 2.2.1 illustrates the interaction between agent and environment.

Since the agent receives partial observations of the environment's state, it has to act under uncertainty. Planning in a POMDP is undecidable, and solving them is often computationally intractable. Approximate solutions are more common, where the agent usually maintains an internal *belief state b* that summarizes its previous experience.

2.2.2 Policies and Value Functions

Most RL algorithms estimate both a *value function* that tells the agent how good it is to be in a given state. [28]

2.2.3 Exploration and Exploitation Trade-off

2.2.4 Sparse Rewards and the Credit Assignment Problem

The (temporal) credit assignment problem (CAP) [22]...

2.2.5 Common Algorithms

2.3 Related Work

2.3.1 Deep Reinforcement Learning

A long-standing challenge in RL is to learn good policies from high-dimensional sensory inputs like vision. Mnih et al. (2013) [25] address this by combining deep learning with RL. A CNN is trained to estimate the

Mnih et al. (2015) [26]...

Hausknecht and Stone (2017) [15] investigates the effects of adding recurrency to a DQN in order to tackle POMDPs. A LSTM is added after the initial

2.3.2 Active Vision

Much of past and present research in machine perception involves a passive observer. Images are passively sampled and perceived. Animal perception, however, is active. We do not only see things, but look for them. One might ask why this is the case, if there is any advantage that an active observer has over a passive one. Aloimonos and Weiss (1988) [2] introduce the paradigm called *active vision*, and prove that an active observer can solve several basic vision problems in a more efficient way than a passive one.

Bajcsy (1988) [**bajcsy_1988**] defines active vision, and perception in general, as a problem of intelligent data acquisition. An active observer needs to define and measure parameters and errors from its scene and feed then back to control the data acquisition process. Bajcsy states that one of the difficulties of this problem is that they are scene and context dependent. A thorough understanding of the data acquisition parameters and the goal of the visual processing is needed. One view lacks information that may be present with multiple views. Multiple views also add the time dimension into the problem.

In a re-visitation of active perception, Bajcsy, Aloimonos and Tsotsos (2018) [**bajcsy_aloimonos_tsotsos_2018**] stress that despite recent successes in robotics, artificial intelligence and computer vision, an intelligent agent must include active perception:

> An agent is an active perceiver if it knows why it wishes to sense, and then chooses what to perceive, and determines how, when and where to achieve that perception

[**bajcsy_aloimonos_tsotsos_2018**]

2.3.3 Visual Search

The perceptual task of searching for something in a visual environment is usually referred to as *visual search*. The searched object or feature is the *target*, and the other objects or features in the environment are the *distractors*. This task has been studied extensively in psychology and neuroscience.

Wolfe (2021) [29] describes a model of visual search

Eckstein (2011) [11] reviews efforts from various subfields and identifies a set of mechanisms used to achieve efficient visual search. Knowledge about the target, distractor, background statistical properties, location probabilities, contextual cues, rewards and target prevalence are all identified as useful. This is motivated with evidence from psychology as well as neural correlates.

Visual search is not always instant, and can in fact often be slow. This is in part due to processing: our visual system cannot process the entire visual field and

Wolfe and Horowitz (2017) [**wolfe_horowitz_2017**] identify and measure a set of factors that guide attention in visual search. One of these is bottom-up guidance, in which some visual properties of the scene draw more attention than others. Another is top-down guidance, which is user driven and directed to objects with known features of desired targets. Scene guidance is also identified, in which attributes of the scene guide attention to areas likely to contain targets.

These works ground the task considered in this project in psychology.

2.3.4 Object Detection

A similar problem can be found in the computer vision literature under *object detection*. The goal of object detection is to, given an input image, detect instances of semantic objects in it. This includes assigning a bounding box to the objects, and classifying the object. The input image is usually passively sampled, and the whole scene is visible at once.

Caicedo and Lazebnik (2015) [6] propose to use deep reinforcement learning for active object localization in images where the object to be localized is fully visible. An agent is trained to successively improve a bounding box using translating and scaling transformations. They use a reward signal that is proportional to how well the current box covers the target object. An action that improves the region is rewarded with +1, and -1 otherwise. Without this quantization, the difference was small enough to confuse the agent. Binary rewards communicate more clearly which transformations keep the object inside the box and which take the box away from the target. When there is no action that improves the bounding box, the agent may select a trigger action (which would be the only action that does not give a negative reward) which resets the box. This way the agent may select additional bounding boxes. Each trigger modifies the environment by marking it so that the agent may learn to not select the same region twice. This is referred to as an inhibition-of-return mechanism, and is widely used in visual attention models [**[16] in caicedo_active_2015**]. This method has a few shortcomings for the problem considered in this project. The object may not be visible in the initial frame so the agent cannot act in the same way.

A separate field is active object search, which is perhaps most closely related to the problem we consider in this work. In active object search,

A similar work by Ghesu et al. (2016) [**ghesu_artificial_2016**] present an agent for anatomical landmark detection trained with DRL. Different from [6] is that the entire scene is not visible at once. The agent sees a limited region of interest in an image, with its center representing the current position of the agent. The actions available to the agent translate the view up, down, left and aright. A reward is given to the agent that is equal to the supervised relative distance-change to the landmark after each action. Three datasets of 891 anatomical images are used. The agent starts at random positions in the image close to the target landmark and is tasked with moving to the target location. While achieving strong results (90% success rate), the scenes and targets are all drawn from a distribution with low variance. Most real-world search tasks exhibit larger variance than anatomical images of the human body.

Zhu et al. (2016) [33] create a model for target-driven visual navigation in indooor scenes with DRL. An observer is given a partial image of its scene as well as an image of the target object, and is tasked with navigating to the object n the scene with a minimal number of steps. The agent moves forwards, backwards, and turns left and right at constant step lengths. They use a reward signal with a small time penalty to incentivize task completion in few steps. They compare their approach to random walk and the shortest path and achieve promising results. This setup is quite similar to the one considered in this report, but the authors make a few assumptions that we do not. They a set of 32 scenes, each of which contain a fixed number of object instances. They focus on learning spatial relationships between objects in these specific scenes, and have scene-specific layers to achieve this. Thus, while they show that they can adapt a trained network to a new scene, their approach is unable to zero-shot generalize to new scenes.

A similar work by Ye et al. (2018) [31] integrates an object recognition module with a deep reinforcement learning based visual navigation module. They experiment with a set of reward functions and find that constant time penalizing rewards can be problematic and lead to slow convergence. Their experiments make the same assumptions as [**zhu_target_driven**] - the scenes and targets used during testing have all been seen during training.

2.3.5 Visual Attention

2.3.6 Coverage Path Planning

[12]

2.3.7 Navigation

The task we consider bears resemblance to many navigation tasks in robotics. Point navigation is the task of ... Object navigation is the task of ... Semantic navigation is the task of ...

Mnih et al. (2016) [24] use a recurrent policy with only RGB images to navigate in a labyrinth. 3D labyrinths are randomly generated, and an agent is tasked with finding objects in them. The same architecture as in [26] is used, but with 256 LSTM cells after the final hidden layer.

Mirowski et al. (2017) [23]...

Henriques and Vedaldi (2018) [17] use a spatial memory...

Gupta et al. (2019) [14] use a latent spatial memory. The They also use a planner that can plan paths given partial information of the environment. This allows the agent to take appearance of visited locations into account when deciding where to look next. The RGB observation is fed through an encoder network that... Planning in this fashion s

Dhiman et al. (2019) [10] critically investigate deep RL for navigation...

Chaplot et al. (2020) [7] build on the idea of an explicit memory by including environment semantics...

2.3.8 Overfitting and Generalization in Deep Reinforcement Learning

Kirk et al. (2021) [21] survey generalization in deep RL.

While deep neural networks have proved to be effective function approximators for RL, they are also prone to *overfitting*. High-capacity models trained over a long time may memorize the distribution seen during training rather than general patterns. While studied in supervised learning, overfitting is generally been neglected in deep RL. Training and evaluation stages are typically not separated. Instead, the final return on the training environments is used as a measure of agent performance.

Zhang et al. (2018) [32] study overfitting and generalization in deep RL. With experiments, they show that RL agents are capable of memorizing training data, even when completely random. When the number of training samples exceeds the capacity of the agent, they overfit to them. When exposed to new but statistically similar environments during testing, test performance could vary significantly despite consistent training performance. The authors argue that good generalization requires that the *inductive bias* of the algorithms is compatible with the bias of the problems. The inductive bias refers to a priori algorithmic preferences, like neural network architecture. When comparing MLPs with CNNs, they find that MLPs tend to be better at fitting the training data are worse at generalizing. When rewards are spatially invariant, CNNs generalize much better than MLPs. The authors advocate for carefully designed testing protocols for detecting overfitting. The effectiveness of stochastic-based evaluation depends on the properties of the task. Agents could still learn to overfit to random training data. For this reason, they recommend isolation of statistically tied training and test sets.

In a similar spirit, Cobbe et al. (2019) [9] construct distinct training and test sets to measure generalization in RL. They find that agents can overfit to surprisingly large training sets, and that deep convolutional architectures can improve generalization. Methods from supervised learning, like L2 regularization, dropout, data augmentation and batch normalization are also shown to aid with generalization.

Hessel et al. (2019) [18] investigate the trade-off between generality and performance from the perspective of inductive biases. Stronger biases can lead to faster learning, while weaker biases potentially lead to more general agents. They find that learned solutions are competitive with domain heuristics, and that they seem to be better at generalizing to unseen domains. For this reason, they argue for removing biases determined with domain knowledge in future research.

Cobbe et al. (2020) [8] introduce a benchmark for sample efficiency and generalization in RL. They make use of procedural generalization to decide many parameters of the initial state of the environment. This forces agents to learn policies that are robust variation and avoid overfitting. To evaluate sample efficiency of agents in the benchmark, they train and test on the full distribution of states. To evaluate generalization, they fix the number of training samples and then test on held out levels. When an episode ends, a new sample is drawn from the training set. Agents may train for arbitrarily many time steps. The number of training samples required to generalize is dependent on the particulars and difficulty of the environment. The authors choose the training set size to be near the region when generalization begins to take effect. Empirically they find that larger model architectures improve both sample efficiency and generalization. Agents strongly overfit to small training sets and need many samples to generalize. Interestingly, training performance improves as the training set grows past a certain threshold. The authors attribute this to the implicit curriculum of the distribution of levels.

2.3.9 Evaluation of Agents

Anderson et al. (2018) [3] discuss problem statements and evaluation measures for embodied navigation agents. They...

Batra et al. (2020) [4] do something similar...

A problem in RL is reproducibility. There is often non-determinism, both in the methods and environments used. This has meant that reproducing state-of-the-art deep RL results is difficult. [16] Henderson et al. (2018) suggest to make future results in deep RL more reproducible.

A similar work by Agarwal et al. (2022) [1] criticises the heavy use of point estimates of aggregate performance. They advocate for a set of performance metrics that take uncertainty in results into account.

3 Method

In this chapter, the method used is described. Section 3.1 formalizes the problem solved. Section 3.2 details the environment used to evaluate solutions. Section 3.3 describes the baseline learning method. Section 3.4 describes the approach used to solve the problem with a learning agent. Section 3.5 describes the experiments conducted to answer research questions 2 and 3.

3.1 Problem Statement

Formally, the searched environment contains a scene with a set of targets. The scene is described by a Euclidean space. In the scene, there are N targets, each described by a subspace. At any given time the agent observes a subspace of the scene, which we call the view. This observation is given in the form of an image. Through a finite set of actions, the agent can transform its view. With a final trigger action, the agent can indicate that there is a target in the view. The goal of the agent is to bring all targets into the view and indicate that they have been found. This is to be done with a minimal number of actions. This corresponds to changing the field of perception.

We denote the task by $\langle \mathcal{M}, \mathcal{T}_0 \rangle$, where $\mathcal{M}$ is a POMDP and $\mathcal{T}_0$ is the probability distribution on the initial states.

Furthermore, we make the assumption that the agent initially has access to its position. With this, it can keep track of which regions are and have been in view before.

3.2 Environment

To train an test an agent for the problem, we use three environments of varying difficulty. In each environment there is a scene with a background of distractors and a foreground of targets. The scenes are drawn from some unknown distribution. The background and foreground are assumed to be correlated. This means that by looking at the background, an agent should sometimes be able to deduce a suitable action. All scenes are assumed to be static in that the actions of the agent do not affect their appearance.

The scenes of each environment are discretized into a grid. We use the same action space, reward signal for all environments. $\mathcal{A} = \{\texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}, \texttt{TRIGGER}\}$ $\mathcal{R}(\int, \dashv) = \ldots$. The

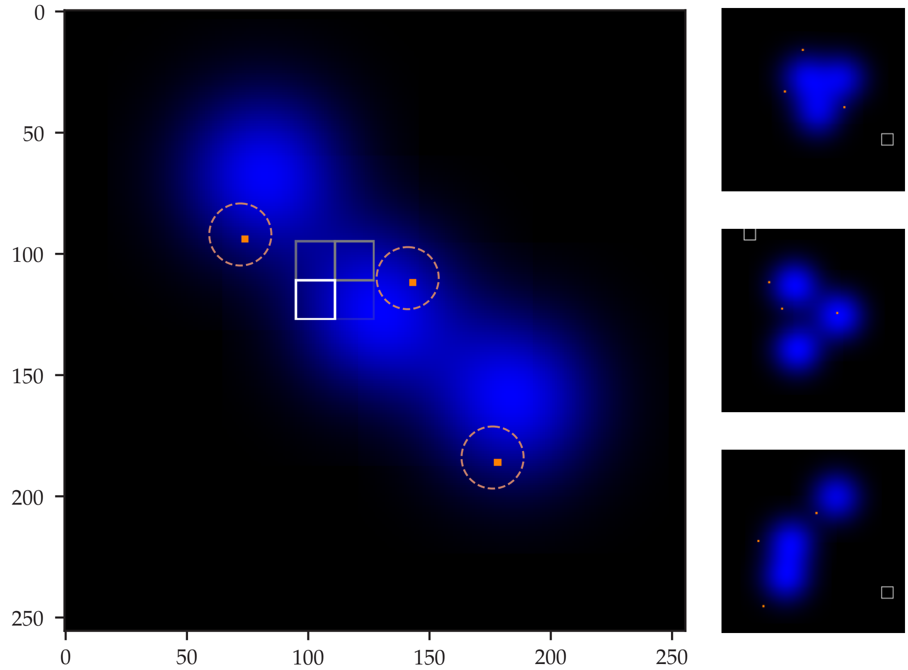

Figure 3.1: Four samples of the first environment. There are three gaussian kernels in the environment, whose height is visualized with the blue channel. There are three targets in the environment, whose location is sampled from the distribution defined by the sum of the three gaussian kernels.

observation space always includes the currently visible region of the environment, an RGB image $x_t \in \mathbb{R}^{3\times W\times H}$.

To incentivize finding targets quickly, the reward signal is set to -1 for each time step. Since viewing a window twice is redundant, such actions are punished by setting the reward to -2. If the agent selects the trigger action when a target overlaps with the window, the reward is set to 5. When all targets have been triggered, or when 1000 time steps have passed, the episode ends.

Averaged over all possible samples, the probability of targets should be uniform over the scene.

The first environment is the simplest environment. The scene is a two-dimensional discrete Euclidean space. The appearance of the scene is given by a 256x256 RGB image. The agent observes a 64x64 sub-image at each time step. In the image there are three Gaussian kernels with random positions. The height of the kernel is indicated by a higher intensity in the blue channel. Targets are 1x1 pixels in the red channel. The locations of the targets are randomized weighted by the height of the Gaussian kernels. This means that the more intense the blue channel, the higher the probability of a target. The idea with this environment is to test that the method learns what we want it to learn. It is easy to determine whether the agent acts well in this environment. Our feeling is that this is something that previous similar works has not done.

The second environment is intended to look like realistic terrain. The environment has a two-dimensional scene whose appearance is given by a 512x512 image. Gradient noise is

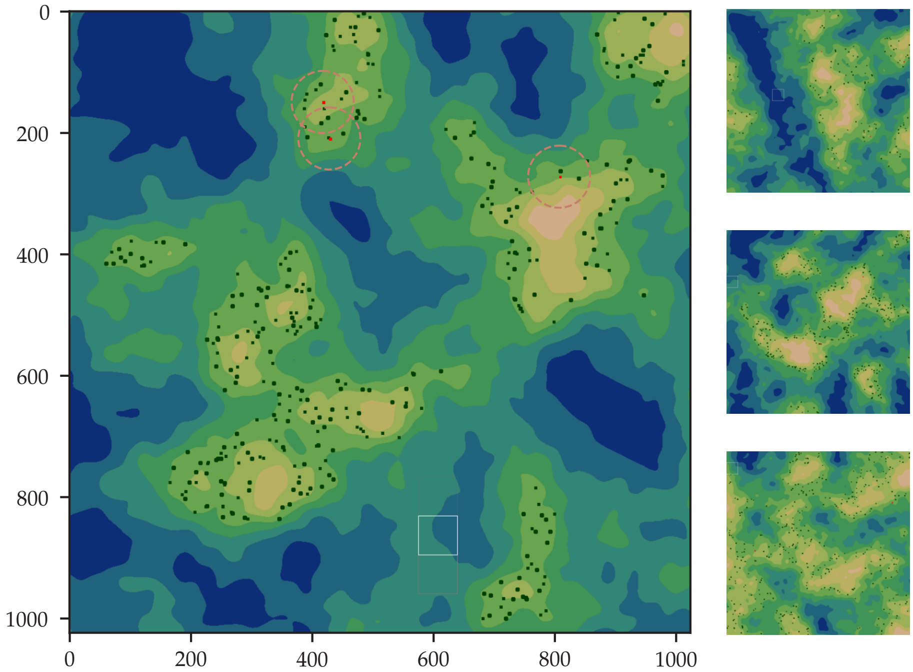

Figure 3.2: Second environment. Terrain seen from above with ocean,

generated and used as a height map. The height map determines the color of the terrain. The height also correlates to the probability of targets. Specifically, targets are located between shores and mountain bases. This environment simulates a UAV search-and-rescue scenario.

The third environment is a three-dimensional version of the second one.

3.3 Baseline

3.4 Approach

We postulate that an effective searcher should be able to:

- Learn a probability distribution of targets and its correlation with the appearance of the scene.
- Avoid visiting the same location multiple times.
- Remember the appearance of previously visited areas in order to prioritize where to go next.

The agent is trained with reinforcement learning using Proximal Policy Optimization with function approximation using neural networks.

3.4.1 Architecture

The architecture of the neural network is presented in Figure X.

The network is split into three main section: the feature extraction, the shared network, and the policy and value heads.

In order to be able to use the same extractor for all environments we resize the RGB image to 64×64 pixels.

3.5 Experiments

The first environment was used to determine a good observation space. By just observing the current window, the agent can never learn a suitable policy to solve the problem. This is because it cannot distinguish between equal windows at different locations. Experiments are run with several additional observation types: window position, ... Results of these experiments are presented for this environment only.

The agent was trained using the algorithm described in Section 3.4 for 100 million time steps in all three environments using PPO, PPG and A2C. Hyperparameters are tuned with random search separately for each environment. For all experiments, the average return per episode is reported together with the theoretically optimal reward (obtained with an optimal path). The agent was trained and tested on the full distribution of environments.

With the best performing algorithm we compare three different reward signals. One gives a constant time penalty of -1 to incentivize finding the target quickly. The second gives a constant time penalty of -2 and an exploration bonus of +1 when a new view is reached. The third gives a reward of +1 for moving closer to the target and a penalty of -1 otherwise, as in [6].

Additionally, experiments to eveluate the generalization capability of the agent were conducted. These were conducted on the procedurally generated terrain environment following the approach suggested in [**procgen**]. During training, the seed pool size was fixed to various sizes to limit the training set size. The agent was trained for varying number of timesteps and then tested on the full distribution of environments. This way, we can get a sense of how much data and simulation is required to use the approach for real-world tasks.

All experiments are conducted on an Intel Core i9-10900X CPU and an NVIDIA GeForce RTX 2080 Ti GPU.

For each experiment, we report the mean return and episode length over time during training. We compare the approach to an exhaustive search and a human searcher with prior knowledge of the characteristics of the searched environments. As per [1], we report results across multiple seeds.

3.6 Implementation

The environment is implemented with Gym [5], and the agent is implemented with PyTorch [27]. Proximal policy optimization was implemented following the official implementation by OpenAI. Some necessary modifications were made to allow for recurrent policies.

4 Results

When it comes to hyperparameters, we find that letting the number of rollout steps be substantially lower than the episode length we achieve much more stable training results. Furthermore, increasing the number of weights in the neural network made it more difficult to train.

We find that the hyperparameters from [procgen] perform well, especially when the number of environments is large.

Also, proximal policy optimization was unstable without reward normalization.

This coupled with a sparse reward signal led to many cases where the agent converged towards a poor local optimum (or perhaps never converged at all).

5 Discussion

This chapter contains the following sub-headings.

5.1 Results

5.2 Method

It is worth considering whether using a learning agent like this is suitable for this task. One could imagine that it is possible to compute an optimal strategy for certain environments. However, this quickly falls apart. The dynamics of environments can vary considerably which may drastically affect how a manual approach is implemented.

Another thing worth discussing is the possibility of combining manual search method with reinforcement learning. One could imagine combining a frontier based approach with a learning approach.

In this work, we have only covered searches where the view is transformed in the spatial domain. However, the method could be applied to a broader category of problems. For instance, one could imagine a scenario when searching along the time dimension is useful. If we let the actions be translations arbitrary translations along the time dimensions in, say, a long audio or video file, the agent could learn to look for landmark features in such modalities.

5.3 The work in a wider context

While automated search systems have many positive uses, like XXX, there are certainly other use cases that could be considered negative. Mass surveillance, XXX, are both very relevant today.

6 Conclusion

Bibliography

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. "Deep Reinforcement Learning at the Edge of the Statistical Precipice". In: *arXiv:2108.13264 [cs, stat]* (Jan. 2022). arXiv: 2108.13264. URL: http://arxiv.org/abs/2108.13264 (pages 9, 13).

[2] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. "Active vision". In: *International Journal of Computer Vision* 1.4 (Jan. 1988). 705 citations (Crossref) [2022-02-07], pp. 333–356. ISSN: 0920-5691, 1573-1405. DOI: 10/cn4mdc. URL: http://link.springer.com/10.1007/BF00133571 (visited on 02/07/2022) (page 6).

[3] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. "On Evaluation of Embodied Navigation Agents". In: *arXiv:1807.06757 [cs]* (July 2018). arXiv: 1807.06757. URL: http://arxiv.org/abs/1807.06757 (page 9).

[4] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. "ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects". In: *arXiv:2006.13171 [cs]* (Aug. 2020). arXiv: 2006.13171. URL: http://arxiv.org/abs/2006.13171 (page 9).

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "OpenAI Gym". In: *arXiv:1606.01540 [cs]* (June 2016). arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540 (page 13).

[6] Juan C. Caicedo and Svetlana Lazebnik. "Active Object Localization with Deep Reinforcement Learning". In: *arXiv:1511.06015 [cs]* (Nov. 18, 2015). arXiv: 1511.06015. URL: http://arxiv.org/abs/1511.06015 (visited on 02/03/2022) (pages 7, 13).

[7] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. "Object Goal Navigation using Goal-Oriented Semantic Exploration". In: *arXiv:2007.00643 [cs]* (July 2020). arXiv: 2007.00643. URL: http://arxiv.org/abs/2007.00643 (page 8).

[8] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. "Leveraging Procedural Generation to Benchmark Reinforcement Learning". In: *arXiv:1912.01588 [cs, stat]* (July 2020). arXiv: 1912.01588. URL: http://arxiv.org/abs/1912.01588 (page 9).

[9] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. "Quantifying Generalization in Reinforcement Learning". In: *arXiv:1812.02341 [cs, stat]* (July 2019). arXiv: 1812.02341. URL: http://arxiv.org/abs/1812.02341 (page 9).

[10] Vikas Dhiman, Shurjo Banerjee, Brent Griffin, Jeffrey M. Siskind, and Jason J. Corso. "A Critical Investigation of Deep Reinforcement Learning for Navigation". In: *arXiv:1802.02274 [cs]* (Jan. 2019). arXiv: 1802.02274. URL: http://arxiv.org/abs/1802.02274 (page 8).

[11] M. P. Eckstein. "Visual search: A retrospective". In: *Journal of Vision* 11.5 (Dec. 30, 2011). 207 citations (Crossref) [2022-02-28], pp. 14–14. ISSN: 1534-7362. DOI: 10.1167/11.5.14. URL: http://jov.arvojournals.org/Article.aspx?doi=10.1167/11.5.14 (visited on 02/22/2022) (pages 2, 7).

[12] Enric Galceran and Marc Carreras. "A survey on coverage path planning for robotics". In: *Robotics and Autonomous Systems* 61.12 (Dec. 2013), pp. 1258–1276. ISSN: 09218890. DOI: 10/f5j2n5 (page 8).

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, Nov. 2016. ISBN: 978-0-262-03561-3 (page 4).

[14] Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. "Cognitive Mapping and Planning for Visual Navigation". In: *arXiv:1702.03920 [cs]* (Feb. 2019). arXiv: 1702.03920. URL: http://arxiv.org/abs/1702.03920 (page 8).

[15] Matthew Hausknecht and Peter Stone. "Deep Recurrent Q-Learning for Partially Observable MDPs". In: *arXiv:1507.06527 [cs]* (Jan. 2017). arXiv: 1507.06527. URL: http://arxiv.org/abs/1507.06527 (page 6).

[16] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. "Deep Reinforcement Learning That Matters". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.11 (Apr. 2018). ISSN: 2374-3468. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11694 (page 9).

[17] Joao F. Henriques and Andrea Vedaldi. "MapNet: An Allocentric Spatial Memory for Mapping Environments". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018, pp. 8476–8484. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00884. URL: https://ieeexplore.ieee.org/document/8578982/ (page 8).

[18] Matteo Hessel, Hado van Hasselt, Joseph Modayil, and David Silver. "On Inductive Biases in Deep Reinforcement Learning". In: *arXiv:1907.02908 [cs, stat]* (July 2019). arXiv: 1907.02908. URL: http://arxiv.org/abs/1907.02908 (page 9).

[19] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735 (page 4).

[20] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1–2 (May 1998), pp. 99–134. ISSN: 00043702. DOI: 10.1016/S0004-3702(98)00023-X (page 5).

[21] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. "A Survey of Generalisation in Deep Reinforcement Learning". In: *arXiv:2111.09794 [cs]* (Jan. 2022). arXiv: 2111.09794. URL: http://arxiv.org/abs/2111.09794 (page 8).

[22] Marvin Minsky. "Steps toward Artificial Intelligence". In: *Proceedings of the IRE* 49.1 (Jan. 1961), pp. 8–30. ISSN: 2162-6634. DOI: 10.1109/JRPROC.1961.287775 (page 6).

[23] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. "Learning to Navigate in Complex Environments". In: *arXiv:1611.03673 [cs]* (Jan. 2017). arXiv: 1611.03673. URL: `http://arxiv.org/abs/1611.03673` (page 8).

[24] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous Methods for Deep Reinforcement Learning". In: *arXiv:1602.01783 [cs]* (June 2016). arXiv: 1602.01783. URL: `http://arxiv.org/abs/1602.01783` (page 8).

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing Atari with Deep Reinforcement Learning". In: *arXiv:1312.5602 [cs]* (Dec. 2013). arXiv: 1312.5602. URL: `http://arxiv.org/abs/1312.5602` (page 6).

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. "Human-level control through deep reinforcement learning". In: *Nature* 518.75407540 (Feb. 2015), pp. 529–533. ISSN: 1476-4687. DOI: `10.1038/nature14236` (pages 6, 8).

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: (), p. 12 (page 13).

[28] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018. 526 pp. ISBN: 978-0-262-03924-6 (pages 5, 6).

[29] Jeremy M. Wolfe. "Guided Search 6.0: An updated model of visual search". In: *Psychonomic Bulletin & Review* 28.4 (Aug. 2021). 29 citations (Crossref) [2022-03-02], pp. 1060–1092. ISSN: 1531-5320. DOI: `10.3758/s13423-020-01859-9` (pages 2, 7).

[30] Jeremy M. Wolfe. "Visual search". In: *Current biology : CB* 20.8 (Apr. 27, 2010). 64 citations (Crossref) [2022-03-02] Publisher: NIH Public Access, R346. DOI: `10.1016/j.cub.2010.02.016`. URL: `https://www.ncbi.nlm.nih.gov/labs/pmc/articles/PMC5678963/` (visited on 03/02/2022) (page 2).

[31] Xin Ye, Zhe Lin, Haoxiang Li, Shibin Zheng, and Yezhou Yang. "Active Object Perceiver: Recognition-Guided Policy Learning for Object Searching on Mobile Robots". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). ISSN: 2153-0866. Oct. 2018, pp. 6857–6863. DOI: `10.1109/IROS.2018.8593720` (page 8).

[32] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. "A Study on Overfitting in Deep Reinforcement Learning". In: *arXiv:1804.06893 [cs, stat]* (Apr. 2018). arXiv: 1804.06893. URL: `http://arxiv.org/abs/1804.06893` (page 8).

[33] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning". In: *arXiv:1609.05143 [cs]* (Sept. 16, 2016). arXiv: `1609.05143`. URL: `http://arxiv.org/abs/1609.05143` (visited on 03/14/2022) (page 7).