Deep Reinforcement Learning of Active Sensing Strategies with POMDPs

Master thesis, by Marcel Zuur

4506626

August 14, 2019

Radboud University

Donders Institute for Brain, Cognition and Behaviour

Artificial cognitive systems

First supervisor: MSc. S.C. Quax

Second supervisor: prof. dr. M.A.J. van Gerven

Abstract

This research aims to investigate the development and performance of artificial neural networks which are capable of extracting task related information from an image using an active sensing strategy. Similar to the way humans deploy an active sensing strategy for the planning of eye movements.

Research suggests that humans can learn such a strategy by strengthening eye movements that have led to the extraction of more valuable information. In that sense the goal of active sensing is to maximize the overall value of extracted information.

By framing active sensing as a Partially Observable Markov Decision Process (POMDP), it can be transformed into an optimization problem suited for Reinforcement Learning. The goal in this setting is to maximize task performance with only limited information. Because the true state of the environment cannot be observed, any task related action must be based on beliefs about the actual state. Accuracy of these belief states can be increased by extracting more valuable information.

Finding an exact solution for this POMDP is intractable and must be approximated. To find good approximations, several recurrent neural networks with different architectures and configurations are trained using Policy Gradient methods.

Training and validating those networks is done using challenging tasks that are designed to cover different important aspects of scene understanding. Since active sensing is a combination of planning focus locations and task performance, these two components will be further investigated. Finally the relevance of these results for understanding human or animal eye movements will be discussed.

Contents

1 INTRODUCTION

We see the world through our eyes and we experience it as seamless, sharp, three dimensional and in color. However, although the field of view of our eyes is fairly large, around 120°for binocular vision, a single eye can only process a small two dimensional part of the visual field with high acuity and in color. This is caused by the anatomy of the eye. When light enters the eye it falls on the retina and only an area of about 2°around the center of, the fovea, has high acuity and full color . Further from the center acuity and the ability to see color degrades fast, with color vision completely gone in the periphery, which starts at around 5°from the center (Saxby, Novemeber 19, 2010).

Viewing an object with high acuity thus means directing the center of the retina towards that object. This causes our eyes to move on average three times a second. Such eye movements are called saccades and the focus periods in between fixations. It is only during those fixations that visual information can be processed by the brain (Rayner, 2009).

What we perceive through our eyes is therefore an integration over time of two dimensional patches, sampled at different locations in space. This sampling is not random, but the result of a strategy of focusing on locations that are of a certain interest. This is process is called selective attention in general, or active sensing when it is about task related sampling and helps the brain by processing only relevant information (Gottlieb, 2012; Gottlieb & Oudeyer, 2018; Yang, Wolpert, & Lengyel, 2016).

Although there has been much research on eye movements, it still remains unclear what cognitive functions are driving them. Earlier models of eye movement have focused mostly on salience, but more recently models using a reinforcement learning paradigm are gaining in popularity. In short these

models propose that given a certain scene, some parts of that scene are more valuable than others and our brain attempts to sample only those locations with the highest value. Over time maximizing the overall value while being limited in the number of possible saccades. Such a strategy can be learned over time by strengthening focusing behaviors that have led to a more positive outcome, e.g. recognizing a certain object, and weakening those that led to a negative outcome. In the brain, the basal ganglia are associated with reinforcement learning and it has been shown that they can automatically bias attention. (Gottlieb, Hayhoe, Hikosaka, & Rangel, 2014; Hayhoe & Ballard, 2014; Lee, Seo, & Jung, 2012).

Framing active sensing as reinforcement learning not only provides us with a plausible learning mechanism in the brain, it also opens the door to a new approach in understanding active sensing namely simulation using artificial intelligence. Reinforcement learning in artificial intelligence uses a similar mechanism in which an untrained system over time learns to behave in such a way that it maximizes rewards received by the environment. Mathematically framed in so-called Markov Decision Processes these artificial models have for instance been capable of learning complex behaviors like playing Atari games, beating humans on the game of Go or learning to cooperate in the game DOTA 2 (Mnih et al., 2015; OpenAI, 2018; Silver et al., 2017).

Although the main focus in most Artificial Intelligence research is not on understanding brain functions or being biological plausible, there are several examples which show that such models can indeed provide insight in how the brain processes information. For instance, it was shown that deep neural networks can be effective predictive models of voxel response in visual areas V1 -V4. It has provided insight in how receptive field properties emerge across the ventral stream. It was even shown that not only are these model capable of capturing essential characteristics of object recognition in space and time, but also in the frequency domain by predicting gamma activity. (Aru & Vicente, 2018; Güçlü & van Gerven, 2015; Kriegeskorte, 2015; Kuzovkin et al., 2018; Yamins & DiCarlo, 2016)

Marblestone, Wayne, and Kording (2016) state that similar to training artificial neural networks, the brain is optimizing cost functions and believe that algorithms for optimization like backpropagation may have correspondences in biological brains. It is even hypothesized that the existence of distinct

visual pathways (dorsal vs ventral) is actually the result of optimizing cost functions of unrelated tasks. Visual guidance of actions versus identification and recognition of objects (Scholte, Losch, Ramakrishnan, de Haan, & Bohte, 2018).

It must be noted that the successful networks so far were all feed-forward networks and that the actual visual cortex is much more complex. For one it does not account for the role recurrent connections may have. Cox and Dean (2014) mention that having recurrent connections might be necessary for incorporating contextual information to enhance otherwise ambiguous inputs. Humans are able to recognize highly degraded images when external context provides additional clues. Kriegeskorte (2015) adds that recurrent networks are more similar to biological neural networks. That the internal state of a recurrent network lends it a memory en enables it to represent recent stimulus history and detect temporal patterns.

Combined, training a recurrent neural network to develop an active sensing mechanism may provide us with more insight in how the brain itself can learn such a strategy.

The research presented in this thesis is the development and analysis of a recurrent neural network that is capable of active sensing. Different models and learning procedures are tested on a number of challenging tasks which require active sensing strategies. The next chapter will start with a background on models of eye movement. Followed by the theory on how active sensing can be modeled as a reinforcement learning problem using the Markov decision process framework.

2 Prior research on eye movements

Only recently is research on eye movement shifting towards active sensing or selective attention paradigms. Prior research was mostly focused on predicting eye movements based using image salience or to answer the question whether humans behave Bayes' optimal in visual search tasks.

This chapter serves as a brief overview of previous research on eye movements. The contribution these models have made to our understanding of eye movements will be discussed, together with their shortcomings.

2.1 Saliency models

In saliency models, overt attention is directed on interesting parts of an image based on low-level features of that image such as high contrast. Bottom-up processing of visual stimuli will detect more salient regions of an image and will direct overt attention to it. To prevent top-down processing, free-viewing is the most suitable task for comparing these models to human performance. Kowler (2011) mention that there is some neuroscientific evidence that spatial patterns of eye movements during inspection of scenes agreed, in general, with computed saliency levels. Research found such patterns of activity linked to voluntary attention or to saccadic planning in cortical areas such as the lateral intraparietal cortext (LIP) or the frontal eye fields (FEF).

2.2 Top down versus Bottom up

Kowler (2011) also mentions that there are several models which state that other factors are far more important than bottom-up features. One class of models, called visibility models, rejects that the goal of eye movements is to focus on what already can be seen. Rather the purpose is to improve the visibility and clarity of eccentric details that cannot be resolved adequately from the current fixation position. These approaches are more interested in for instance how eye movements can improve task performance.

Tatler, Baddeley, and Gilchrist (2005) tried to resolve the debate about the contribution of low level bottom-up processing (salience) versus top-down processes. It was found that consistency in fixation locations selected by observers decreased after the first few fixations after stimulus onset. It was argued that salience does not change during viewing, but that the increase in variability was caused by the different top-down strategies used.

Wolfe and Horowitz (2017) also studied the effect of top-down and bottom-up effects on eye-movements and came up with five forms of guidance, with salience being one of them:

1. Bottom-up, stimulus driven guidance
2. Top-Down, user driven guidance. Attention is directed to objects with known features of desired targets
3. Scene guidance. Attributes of the scene guide attention to areas likely to contain targets.
4. Guidance based on the perceived value of some items or features.
5. Guidance based on the history of prior search.

The importance of top-down processing is also acknowledged by Koehler and Eckstein (2017) which state that object information drives perceptual decisions and eye-movement behavior more than background information. A conclusion supported by Kowler (2011) who mentions that saccades on average land near the center of target shapes.

Brain research done by Groen, Silson, and Baker (2017) shows that there is evidence for dynamic interactions between low- and high-level representations over time and that it is likely that all of this information can be useful for some aspects of scene perception.

2.3 Meaning

An interesting approach is the concept of meaning maps by Henderson and Hayes (2017). Similar to saliency maps, interesting regions of an image are highlighted, but by semantic meaning rather than low-level features. The authors argue that very early after viewing a scene, the gist can be extracted and will activate predictions of what is likely to be present and where it will be located. These initial representations can then be used to generate predictions for guiding attention to regions that have not yet been identified. In a follow-up Henderson and Hayes (2018) found that humans are highly sensitive to the distribution of meaning in visual scenes from the earliest moments of viewing. They conclude that attention is primarily controlled by knowledge structures that contain information about the likely semantic context and spatial distribution of that content in a scene.

2.4 Ideal observer models

Najemnik and Geisler (2005) show that humans eye movements are nearly Bayes optimal in a visual search task. They constructed an ideal Bayesian searcher which maximizes the probability of correctly localizing the target after the next fixation is made and the posterior probabilities are updated. This ideal search had full knowledge of the image statistics and was therefore able to determine the eye movements that lead to the most information about the target location. Humans outperformed all model with random eye movements, and almost reached the performance of the ideal searcher.

However a real challenge with such ideal observer models is how they can be used for naturalistic tasks. These models require the relevant statistics to be known and it is unsure how these must be

characterized in natural tasks. Even with all statistics known it will be a challenge to derive an ideal observer model of it (Geisler, 2011).

In a follow-up study, Najemnik and Geisler (2009) acknowledged that the Bayesian model is limited in its uses and that it is highly unlikely that the brain is actually performing Bayesian inference. To overcome this, they created a simpler heuristic that could be implemented by a biological nervous system. Their so-called entropy limit minimization (ELM) searcher performed as well as the Bayesian ideal searcher on the same search tasks from the previous study. The authors concluded that the brain might not need complex models to behave optimal, but that simple rules can lead to optimal fixation selection.

2.5 Active Sensing

What all the models described above have in common is that they are treating visual perception as an isolated process. In active sensing models it is assumed that perception and action are coupled processes. When humans perform a task, the goals of perception is to make inferences about the information it is receiving from the sensory organs, while actions are needed to focus those sensory organs to efficiently extract task-relevant information. When translated to eye movements this would mean a constant interplay between perceiving task-related information and deciding on where to focus next for valuable information (Yang, Wolpert, & Lengyel, 2016).

Using this paradigm, Yang, Lengyel, and Wolpert (2016) crafted a task in which participants were asked to discriminate between a patchy and a stripy pattern, resembling either the fur pattern of a cheetah or a zebra. Because deciding between the two can be a matter of life and death, it is of the highest importance to have a sampling strategy that can quickly identify the correct pattern. Because the two presented patterns had known statistics, it was possible to create a Bayesian Ideal Observer.

By comparing the performance of the Bayesian Ideal Observer to the fixation patterns human participants were using when asked to identify the patterns, it was found that humans likely plan eye movements with the goal of maximizing information.

A disadvantage of the approach by Yang, Lengyel, and Wolpert (2016) is that to create their Bayesian Ideal Observer it is necessary that all the stimulus statistics are know. For the patchy and stripy patterns, which were create using Gaussian Processes, this was no issue. However applying this approach on more complex natural stimuli is far from trivial.

Fortunately the theory behind active sensing lends itself to reformulate it as an optimization problem that can be approximated using Reinforcement Learning. This can be achieved by changing the goal of maximizing information to maximizing future reward and using the information gain associated with sampling at a certain location as reward.

The next chapter will go into much more depth about how to transform active sensing to such an optimization problem.

3 Learning active sensing strategies through Reinforcement Learning

One of the most appealing aspects of active sensing, is that it can be seen as something which can be learned through reinforcement learning. Several authors suggest that the brain has a valuation system that plays a key role in determining to which features focus must be shifted to. The value of an eye movement lies in reducing uncertainty and increasing the expected reward of a future outcome (Balcarras, Ardid, Kaping, Everling, & Womelsdorf, 2016; Gottlieb, 2012; Gottlieb & Oudeyer, 2018; Lee et al., 2012).

This chapter serves as a theoretical foundation for modeling active sensing as a Partially Observable Markov Decision Process (POMDP), which is an optimization problem that can be approximated using a Reinforcement Learning algorithm.

3.1 Perception as hypothesis testing

Perception can be regarded as a form of hypothesis testing. (Friston, Adams, Perrinet, & Breakspear, 2012; Gregory, 1980). After receiving sensory input $x \in X$, our brain tries to find the best explanation for what is causing x by fitting various hypothesis $h \in H$ about x under model M. This model represents everything the brain knows about what can be generating or influencing x. For instance perceiving the same object under changing lighting conditions will give different visual sensations, however the best

explaining hypothesis should not change because of that. Thus the model M determines the shape of the probability function used for testing.

Using a statistical procedure called a Maximum Likelihood Estimate or MLE we can find the hypothesis h for which it is the most likely that x will occur under model M

$$\underset{h \in H}{\operatorname{argmax}} P(x \mid h, M) \tag{3.1}$$

A drawback of this procedure is that it does not take into account any prior knowledge you may have about the likelihood of the hypothesis itself given M. For instance, perceiving snow in the summer may have a high likelihood when you are in a polar region, but not at the equator.

By using Bayes' rule, it becomes possible to incorporate such prior knowledge and calculate a Maximum a Posteriori estimate or MAP.

$$\underset{h \in H}{\operatorname{argmax}} \frac{P(x \mid h, M)P(h)}{P(x, \mid M)}) \tag{3.2}$$

The denominator in equation 3.2 is called the model evidence and is used for normalization. It captures the probability of observing x and is intractable, since it needs the calculation of every possible combination of x under M. Since the model evidence is only used for normalizing, it can be left out of the equation when you are only interested in the MAP estimate. The model variable M is fixed and for convenience it can be implicitly assumed.

$$\begin{aligned} P(h \mid x) &\propto P(x \mid h)P(h) \\ \text{MAP} &= \underset{h \in H}{\operatorname{argmax}} P(x \mid h)P(h) \end{aligned} \tag{3.3}$$

3.1.1 Perception is not a static process

So far, perception was described as a static process, where sensory input x provides all the information for fitting h. In reality perception is an active process where actions, for instance eye movements,

influence perception and what is perceived will influence any next action to be taken. Also the sensory systems are heavily restricted in how much of a scene can be perceived at once.

Most of the time you will need multiple observations at different locations in space to sample enough sensory data for reliable hypothesis testing. Given the reciprocal relationship between perception and action, each next sampled x_{t+1} will be the result of all the previously sampled data $x_{0:t}$. Where x is the result of direction the eye to a certain region in space.

To reflect this in equations 3.1 and 3.3, x should be replaced by latent state z.

$$z_{t=0:t=T} = \{x_{t=0}, x_{t=1}, \ldots, x_{t=T}\} \tag{3.4}$$

This notation implies that any organism that is capable of active sensing should have a memory that can remember previous states and a mechanism that can integrate over this sequence of states to form latent state z.

Second it is necessary to have a method for selecting the best actions given z. Good actions should lead to a more desirable state. For instance finding a good meal during foraging for food or solving a puzzle. Bad actions will leave the organism in a more negative state. In that sense, acting will lead to reward, either negative or positive. Since positive rewards reflect desired outcomes, a good objective for any organism would be to maximize overall reward. (Sun, Gomez, & Schmidhuber, 2011; Yang, Lengyel, & Wolpert, 2016; Yang, Wolpert, & Lengyel, 2016)

3.2 Active sensing through reinforcement learning

It is possible to create and train a neural network that is capable of learning active sensing. This can be done through reinforcement learning, since this shares the same goal, namely maximizing overall reward.

Examples like the the "Recurrent Attention Model" (RAM) developed by Mnih, Heess, and Graves (2014) or the DRAM model created by Ba, Mnih, and Kavukcuoglu (2014) already demonstrate that it is possible to train a neural network to select only relevant parts of an image for the given task.

3.2.1 Maximizing rewards

If you consider time to be discrete with equal time-steps t overall reward is nothing more than the sum of rewards collected at each t.

$$R = \sum_{t=0}^{\infty} r_t \tag{3.5}$$

Theoretically, overall reward can be maximized by always choosing the action a_t that will lead to the maximum reward possible. Of course this is an impossibility, since the organism should have complete a priori knowledge about what the maximum rewards is and how to achieve it. In reality it should predict what the future outcome will be. Predicting future outcomes is something the organism should learn and continuously improve. Thus actions should be taken on latent state z and expected outcome R_t. More formally an organism should learn a policy $\pi(z, a)$ which aims to maximize expected total reward. A common notation is to use $Q^{\pi}(z, a)$ as the function for the expected future reward when observing latent state z and taking action a under policy π. This is known as the state action value function and is given by

$$Q_{\pi}(z, a) = \mathbb{E}\left[\sum_{k=1}^{\infty} r_{t+k} \mid z_t = z, a_t = a\right] \tag{3.6}$$

Always choosing the highest Q-value using an optimal Q-function will always lead to the optimal policy.

$$\pi(z, a) = \underset{a}{\operatorname{argmax}}\, Q(z, a) \quad \forall z \in Z \tag{3.7}$$

3.2.2 Markov Decision Process

A useful mathematical framework for modeling decision making is the Markov Decision Process or MDP. It is based on the Markov property, which states that the future is independent of the past given the present. Given a fully observable state s at time t this is given by the following relation:

$$P(s_{t+1} \mid s_t) = P(s_{t+1} \mid s_1 \dots s_t) \tag{3.8}$$

Assuming the Markov Property, the state should capture all relevant information from the history. Note that this property does not hold for the latent state z defined earlier, because it represents the integration over observations of an unknown full state s. For correctness the letter z will be replaced by s. Later in this chapter it will be discussed how partially observed states can be incorporated in a Markov Decision Process.

The probability of transitioning from one state s to some other state s' is given by the state transition probability function.

$$P_{ss'} = P(s_{t+1} = s' \mid s_t = s) \tag{3.9}$$

This is a probability distribution over possible successor states. A finite set of states S with the Markov property and a state transition function P. Finding this distribution can be achieved with a so-called Markov process or Markov Chain where the next state is randomly selected from the total set of states.

Besides knowing the possibility of reaching a certain state, we should also know what reward we will receive when transitioning to the next state. For this we can extend the Markov chain with rewards. This is defined as a tuple $< S, P, R, \gamma >$ With R being the reward function $R_s = \mathbb{E}[r_{t+1} \mid s_t = s]$ and discount factor $\gamma \in [0, 1]$

Since our objective should is to maximize total future our reward function should reflect that. For this the return G_t is used, which is the total discounted reward, starting from time-step t

$$G_t = \sum_{k=1}^{\infty} \gamma^k R_{t+k} \tag{3.10}$$

Using the discounted return, the value of a state is the expected return starting from state s. The discount factor serves two main purposes. First it prevents the return from going to infinity. This can happen when the Markov Process is cyclic. Another good reason to use a discount factor, is that it gives less weight to similar rewards in the more distant future. For instance receiving the same amount of money today or in two weeks would be equal in reward without the discount factor. In most cases it should be more beneficial to receive the same reward sooner. The discount factor reflects this by adding less weight to the more distant reward.

$$V(s) = \mathbb{E}[G_t \mid s_t = s] \quad \forall s \in S \tag{3.11}$$

The value of a state can be decomposed in the immediate reward r_{t+1} and the discounted value of the successor state $\gamma V(s_{t+1})$. Using this distinction between immediate and future rewards, we get the Bellman equation for Markov reward processes.

$$\begin{aligned} V(s) &= \mathbb{E}[R_{t+1} + \gamma V(s_{t+1}) \mid s_t = s] \quad \forall s \in S \\ &= R_s + \gamma \sum_{s' \in S} P_{ss'} V(s') \end{aligned} \tag{3.12}$$

Extending the Markov Reward Process with decisions gives tuple $< S, A, P, R, \gamma >$ and is known as a Markov Decision Process. A is a finite set of actions. As mentioned in the previous section, a

policy defines which action should be taken when observing a state. A policy is defined as a probability distribution over actions, given states.

$$\pi(a, s) = P(a_t = a \mid s_t = s) \tag{3.13}$$

There exists a simple relationship between $V(s)$ and $Q(s, a)$ The value of a state is the value associated with the maximum Q value over all possible actions.

$$V(s) = \max_a Q(s, a) \quad \forall s \in S \tag{3.14}$$

Similar to the Bellman equation for state value functions, the state-action value function can also be decomposed in immediate and future rewards.

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}[R_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a] \\ &= R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_\pi(s') \end{aligned} \tag{3.15}$$

It shows the distinction between the immediate reward of taking action a after observing state s and the expected discounted future rewards multiplied with the probability of transitioning to state s'

Under policy π the expected value of a state $V_\pi(s)$ is defined by

$$V_\pi(s) = \sum_{a \in A} \pi(a, s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_\pi(s') \right) \tag{3.16}$$

And similar the expected state action value $Q_\pi(s, a)$ by

$$Q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a', s') Q(s', a') \tag{3.17}$$

This combines the probability of choosing action a under π with the transition probability of moving from s to s' and the immediate reward after taking action a and the expected future reward when following policy π for future actions.

Since an optimal policy will always choose the action with the highest value, the optimal policy is deterministic.

$$\pi(s,a) = \begin{cases} 1 & \text{if } a = \text{argmax}_{a \in A} Q(s,a) \\ 0 & otherwise \end{cases} \tag{3.18}$$

With that we end up with the Bellman optimality equation.

$$\begin{aligned} Q(s,a) &= R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V(s') \\ &= R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} Q(s',a') \end{aligned} \tag{3.19}$$

With the equations described in this section, we have the basis for learning the value of states and how to behave optimal. Since you can only observe immediate rewards and can only anticipate on future rewards the distinction between the two is a natural one. By remembering the rewards associated with the transitions from $s \to s'$ and action a' the state-action values can be updated to form better predictions of the future. Therefore learning in this MDP setting is done by acting in the environment, receiving immediate rewards and updating $Q(s,a)$ for better future prediction.

3.2.3 Belief MDP

In the MDP framework, it is assumed that states are fully observable and contains all the necessary information from the past. It is a necessity resulting from the Markov Property. In the active sensing setting choosing actions is not based on a single state, but on a history of partially observed states. Which means that for the individual observed states, the Markov Property will not hold. Also it is the

integration of a sequence of states that will provide the necessary information to act on and not the individual states.

For this special case, the MDP can be extended to form a new tuple of seven $< S, A, O, P, R, Z, \gamma >$. The two new elements are O, which is a finite set of observations and Z, which is an observation function.

$$Z^a_{s'o} = P(o_{t+1} = o \mid s_{t+1} = s', a_t = a) \tag{3.20}$$

It defines the probability that observation o will be recorded when action a is taken and the transition to s' has been made. This extension is called a Partially Observable Markov Decision Process (POMPD). It describes the situation in which the true state of the environment cannot be observed directly, but must be approximated through sensory measurements. Since the true state cannot be used for decision making, the best alternative is to choose a_{t+1} based on the sequence of observations that led to the current observation o_t. Since observations are the result of acting on rewards, history can be regarded as a sequence of actions, observations and rewards.

$$H_t = a_0, o_1, r_1 \ldots a_{t_1, o_t, r_t} = o_0, \ldots o_t \tag{3.21}$$

It is infeasible to keep track of a complete history of observations. Instead a belief state is formed, which is a probability distribution over being in state s conditioned on the history

$$b(s) = P(s \mid H) \quad \forall s \in S \tag{3.22}$$

Since the above equation still contains the full history, but we want a method of updating $b'(s')$ based purely on the current belief state b, action a and observation o. This diminishes the need to keep track of history H, but more important it would make b compliant with the Markov Property.

$$\begin{aligned} b'(s') &= P(s' \mid o, a, b) \\ b'(s') &= \frac{P(o \mid s', a, b)P(s' \mid a, b))}{P(o \mid a, b)} \\ b'(s') &\propto P(o \mid s', a, b)P(s' \mid a, b) \\ b'(s') &\propto Z^a_{s'o} P(s' \mid a, b) \\ b'(s') &\propto Z^a_{s'o} \sum_{s \in S} P(s' \mid a, b, s)P(s \mid a, b) \\ b'(s') &\propto Z^a_{s'o} \sum_{s \in S} P^a_{ss'} b(s) \end{aligned} \tag{3.23}$$

With the belief states being Markovian, we can now write the POMPD as a belief MDP. This is a tuple $< B, A, P, R, \gamma >$. The only difference with a normal MDP is that the set of fully observable states S is now replaced with a set of belief states B

This changes the optimal Bellman state-action value function to

$$Q_\pi(b(s), a) = R^a_{b(s)} + \gamma \sum_{b(s)' \in B} P(b(s') \mid b(s), a) V(b(s')) \tag{3.24}$$

3.2.4 Q Learning

For an MDP with only a limited set of state-action transitions, it is possible to get an exact solution for the optimal Bellman state-action value function. Solving required creating a Q-table where the rows represent states and the columns actions. Each cell then represents the Q value for that state-action pair. After exploring every possible combination of states and actions, we will end up with a Q-table that reflects the complete MDP. With all Q-values known it becomes possible to find the optimal policy following equation 3.18

For larger problems it is unfeasible or even impossible to keep track of every possible transition. Especially in the case of continuous action or state spaces using a Q-table is impossible. A common solution is to use a neural network as a function approximator for $Q(s, a)$. Such a neural network is trained supervised by minimizing a loss function using a gradient descent procedure. Basically it is performing a regression towards the true Q value. $\left\| Q_\pi(s, a) - \hat{Q}(s, a) \right\|_2^2$. Since the true Q-function cannot be known, it must be substituted by a target state value function.

The value for the target depends on which class of learning methods is chosen. Two main classes are Monte Carlo methods and Temporal Difference Methods (Sutton & Barto, 2018). Using a Monte Carlo method involves following the current policy for one episode and storing all action-state transitions and received rewards. $Q_\pi(s_t, a_t)$ is then estimated by averaging over the return R_t, see equation 3.10. In a temporal difference method, values are updated using a bootstrap procedure. In its most simple variant, called TD(0), values are updated after each time-step using the reward received at $t + 1$. Which gives the target $r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$

3.3 Policy gradient

In the previous section it was explained how can frame active sensing as a belief MDP and how this can be learning through the Q-learning method. Although this method has the property that at least in theory it can be completely solved, it has the disadvantage of needing a very large number of samples to learn a good policy. This results in a long training time and even the possibility of not being able to learn a good policy when you only have a limited number of samples. A alternative to Q-learning that tries to solve these problems is the Policy Gradient Method.

A policy gradient method strives to find an optimal policy directly from it's environment. This in contrast to Q-learning methods, where the optimal policy is derived from the Q-function.

A policy gradient method is basically an optimization problem where the goal is to optimize the policy parameters θ such that the expected return is maximized. Based on a trajectory of length T we have the following objective function.

$$J(\theta) = \mathbb{E}_{\pi\theta}[\sum_{t=0}^{T-1} \gamma^t r_t] \tag{3.25}$$

Where γ is an optional discount parameter for giving less weight to future rewards. This objective function can be optimized by using a gradient ascent procedure.

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta) \tag{3.26}$$

The intuition behind the procedure is that the overall reward is maximized by iteratively adjusting the parameters θ with small steps in the direction of the policy gradient $\nabla_\theta J(\theta)$. The step size is defined by the parameter α and the policy gradient is given by.

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} \tag{3.27}$$

3.3.1 Score function

The policy gradient can be estimated using the log-derivative trick, which is shown below (Williams, 1992). For simplicity $f(x)$ is used for the function to optimize, which would be the reward function in this context.

$$\begin{aligned}
\nabla_\theta \mathbb{E}[f(x)] &= \nabla_\theta \int p_\theta(x) f(x) dx \\
&= \int \frac{p_\theta(x)}{p_\theta(x)} \nabla_\theta p_\theta(x) f(x) dx \\
&= \int \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)} p_\theta(x) f(x) dx \\
&= \int p_\theta(x) \nabla_\theta \log p_\theta(x) f(x) dx \\
&= \mathbb{E}[f(x) \nabla_\theta \log p_\theta(x)]
\end{aligned} \tag{3.28}$$

The last line in equation 3.28 now consists of two terms which both can be determined using a sampling strategy. Using a Monte-Carlo sampling method, we sample roll-outs of length T from the environment under the current policy $\tau \sim \pi_\theta(s, a)$. The first term, the reward function to optimize then becomes $f(x) = \mathbb{E}[R(\tau)]$. The second term is called the score function and can be estimated using

$$\nabla_\theta \log p_\theta(x) = \nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t \mid s_t) \tag{3.29}$$

Combining the log-derivative trick and Monte-Carlo sampling we get the following for estimating the policy gradient.

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau) \nabla_\theta \left(\sum_{t=0}^{T-1} \log \pi_\theta(a_t \mid s_t) \right)] \tag{3.30}$$

Most optimal reward function would be the Q-function. $Q^\pi(s_t, a_t) = \mathbb{E}[r_t + r_{t+1} \dots r_{T-1}]$

Directly updating the parameters with $\theta_{t+1} = \theta_t + \alpha \nabla_\theta \mathbb{E}[\tau \sim \pi_\theta[r(\tau)]$ suffers from high variance and is not usable in practice.

3.3.2 Baseline

A common solution to reduce variance is to subtract a baseline $b(s)$ function from the reward function. The most optimal baseline function would be the expected return using the state-action value function.

$$b(s_t) \approx \mathbb{E}[r_t + r_{t+1} \dots r_{T-1}] = Q^{\pi}(s_t, a_t) \tag{3.31}$$

It might be more appropriate to give less weight to future rewards. Using a discount factor γ the baseline function changes to

$$b(s_t) \approx \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \dots \gamma^{T-1-t} r_{T-1}] = Q^{\pi,\gamma}(s_t, a_t) \tag{3.32}$$

Using a discount factor and a baseline, we get the the following policy gradient estimation.

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(\sum_{t'}^{T-1} r_{t'} - b(s_t)\right)\right] \tag{3.33}$$

As can be seen from the equations above, the ideal baseline function would be the Q-function. This same Q-function would also be the most optimal reward function. So the expected value of $\mathbb{E}_{\tau\sim\pi_\theta}[b(s_t) - r(s_t)] = 0$. Meaning that this baseline is an unbiased estimate of the true reward function.

In practice it is not possible to get an exact value for the true Q-function (or reward function) and therefore it must be approximated by a function approximator $Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$. This can be done by training a neural network to output the approximated Q-value. This type of network is commonly known as a critic network. It can be trained supervised by regressing towards the reward function by minimizing $\|Q^{\pi_\theta}(s_t, a_t) - Q_w(s_t, a_t)\|_2^2$

Although the exact Q-function would be the most optimal baseline and reward function, it can be hard to approximate. Certainly for high-dimensional or continuous action spaces, a poorly fitted

Q-function may introduce a large bias and prevent learning a good policy (Schulman, Moritz, Levine, Jordan, & Abbeel, 2015; Tucker et al., 2018).

A common strategy is to use the discount state-value function $V^{\pi_\theta,\gamma}(s)$ instead of the Q-function as a baseline. The state-value function is an unbiased estimate of the Q-function.

$$\begin{aligned} V^{\pi_\theta}(s) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \mid s_0 = s] \\ &= \mathbb{E}_{a\sim\pi}[Q^{\pi_\theta,\gamma}(s,a)] \end{aligned} \tag{3.34}$$

$$A^{\pi_\theta,\gamma}(s,a) = Q^{\pi_\theta,\gamma}(s,a) - V^{\pi_\theta}(s) \tag{3.35}$$

Equation 3.35 shows the advantage function. This function can be thought of as representing how much better an action was than expected. Using an advantage function instead of the reward function means that the gradient ascent procedure will now take gradient steps only in the direction of actions that actually improve the policy. This in comparison to using the reward function, which will produce gradient updates for all rewards.

A policy gradient method utilizing the advantage function is commonly called an Advantage Actor Critic (A2C) algorithm. As mentioned before, getting the exact Q-value is impossible and can only be approximated. This implies that reward functions and value functions can only be approximated as well.

The advantage function is estimated by $\hat{A}_t = \hat{R}_t - V^{\pi_\theta}(s_t)$. The value function can be learned by minimizing $\|R(s_t) - V^{\pi_\theta}(s_t)\|_2^2$

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(\sum_{t'}^{T-1} A^{\pi_\theta,\gamma}(s_t,a_t)\right)\right] \tag{3.36}$$

Such a setup could be implemented using just two neural networks. One for learning the policy (actor) and the other one for the value function (critic). The policy network should approximate an appropriate differentiable probability distribution. For instance when using a policy with discrete

actions, the final layer of the neural network should be the output of a softmax function over the possible actions. The probability distribution is the categorical distribution. For continuous actions a continuous probability distribution should be used and the Gaussian distribution is the most natural choice. The output of the actor network should be the parameterization of the Normal distribution for every action dimension. It should at least output μ, where σ^2 can be either a fixed value or be a trainable variable as well.

3.3.3 Exploration-Exploitation

Independent of which class of learning methods is chosen, the true MDP is unknown and must be learned by exploring the environment. One approach would be to have a policy that always randomly determines which action should be chosen. Under the assumption that every state is reachable through a random process, over time every state-action transition will be visited and used for improving the state-value approximation. Similar to the table approach, large state and action spaces will give a large set of combinations that all have to be visited. Continuous state and or action spaces will be even more problematic. The complete opposite of a random policy is a deterministic policy, such as the optimal policy defined in equation 3.7. As the approximation of $Q(s, a)$ reaches the true function it might be more beneficial to have a policy that follows the optimal path rather than a random one.

There is a trade-off. A random policy is guaranteed to visit every state, whereas a deterministic policy will only visit the states based on it's decision-rule. Therefore a random policy may take a very long time before it has visited all possible combinations of states and actions, while a deterministic policy will never explore alternatives. What's usually done in the learning phase is having a policy that combines both. A popular method that can be used with discrete action spaces is ϵ-greedy. The ϵ parameter determines the probability of choosing a random action over a deterministic action.

$$a_t = \begin{cases} \text{argmax}_{a \in A} Q(s, a) & \text{with probability } 1 - \epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases} \tag{3.37}$$

In continuous action spaces common methods include adding Ornstein Uhlenbeck noise to the chosen actions or sampling the actions from a stochastic policy, e.q. a Gaussian distribution (Lillicrap et al., 2015; Uhlenbeck & Ornstein, 1930; van Hasselt & Wiering, 2007).

Whatever method is chosen, the dilemma is always choosing between speed and finding better alternatives.

4 Methods

As mentioned in section 3.2 two existing models, the ”Recurrent Attention model” RAM and”Deep Recurrent Attention Model” DRAM, already demonstrated that it is possible to learn an attention strategy through reinforcement learning. (Ba et al., 2014; Mnih et al., 2014). These two models will be tested against two new models.

The RAM and DRAM models will serve as base for the new models, but since those models were not developed for being biologically accurate several enhancements were made to improve on that. There also seems to be room to improve the existing models and several options will be discussed.

To test the four models and their possible enhancements, a number of new experiments were developed that are inspired by the experiments done in the RAM and DRAM papers, but should be more challenging and cover more aspects of active sensing.

4.1 Existing RAM and DRAM models

Since the already existing RAM and DRAM models will serve as base models for all the experiments done in this research, this section will discuss in more detail from which components those models are made-up and what their purpose is. The DRAM model is actually an extension of the RAM model and therefore this section will start with a description of the RAM model, while for the DRAM model only the parts that differ will be discussed.

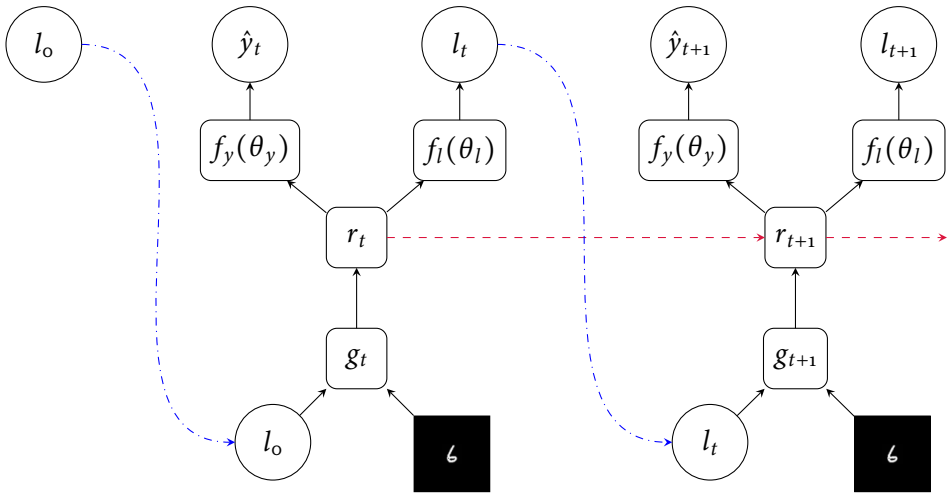


Figure 4.1. Recurrent Attention Model (RAM) with:

l_o = initial sampling location.
Image = full image from which a patch is extracted at location l_t.
g_t = state of the glimpse network at time t, receives both input current location as the extracted patches.
r_t = core network, a recurrent layer which forms the belief state at time t.
$f_y(\theta_y)$ = action network.
$\hat{y}_t$ = output of the action network, class labels in the context of the presented research.
$f_l(\theta_l)$ = policy network.
l_t, l_{t+1} = output of the policy network, which is the next sampling location.

4.1.1 Recurrent Attention Model

A schematic diagram of the RAM model can be seen in figure 4.1. The RAM model operates by defining a fixed number of glimpses after which the action network is asked to which class the presented stimulus belongs. Other parameters are the size of the viewing window, called the glimpse sensor in the RAM paper, the number of allowed zoom-levels and the standard deviation of the Gaussian distribution from which the next glimpse locations are sampled.

Glimpse Network

The Glimpse Network g_t is the part of the model that extracts and processes image patches from the full image presented to the network. Starting at an initial location l_o, which is a two-dimensional coordinate from which the X and Y component are sampled from a uniform distribution with range $(-1, 1)$. Coordinate $(0, 0)$ corresponds to the center of the image, $(1, 1)$ to the bottom right and $(-1, -1)$ top the top left.

At each coordinate l_t a rectangular patch with a predefined shape is extracted from the full image. To mimic the effect of having a fovea with high resolution and low viewing angle combined with a periphery that has a much lower resolution but a high viewing angle, extra zoom levels can be specified. For instance with two zoom levels, the first image is extracted using the given shape, the second is a patch of double size. However, this patch is downscaled to have the same resolution as the first patch. The patches are then concatenated horizontally. With each extra zoom level this doubling and downscaling is repeated.

In the most simple form, the glimpse network g_t actually consists of three fully-connected layers. The first layer receives l_t, the second layer the concatenated image patches and the third receives the summed output of those two layers. The end result is a representation of the extracted patches (What), combined with it's location in space (Where)

For experiments having stimuli with not that many details having just three fully-connected layers is sufficient, for more high detailed images like natural images extra convolutional layers can be added to process the extracted patches.

Core network

The core network is a recurrent layer, which takes the glimpse representation g_t as input. With each time-step it receives a new g_t to update its belief state. In the original RAM paper the most simple form for a recurrent layer is used, namely using the previous output at t as input at $t + 1$. However this approach can make the training procedure unstable because of exploding gradients (Pascanu, Mikolov, & Bengio, 2013).

Instead Long Short-Term Memory (LSTM) cells are used, which should reduces the risk of having an unstable network. However in practice the exploding gradient problem was still present and therefore an extra mechanism was added to prevent this, namely gradient clipping. The clipping value is determined empirically by keeping track of the global norm of the gradients and determining above which value the model became unstable. As soon as the global norm exceeds the manual decided

clipping value, gradients are clipped so that their values do not exceed the clipping value. This does mean that individual gradient values can be larger than the clipping value as long as the global norm stays low enough.

Action network

At each t, the core network will update it's belief state and send it's output to the action network. For every t, with $t = 1, t = 2 \dots T$ the action network will process this belief state using a single fully-connected layer. The output of this network after training should be the correct action for the given belief state. Action in this sense, is a general term for whatever action is appropriate for the task the network is performing. Since all experiments done in this research are classification tasks, the action corresponds to selecting the correct class label.

Although classification performance is only measured by calculation the fraction of correct predictions at time $t = T$, the action network is called at each $t > 0$ to collect rewards. The rationality for this is that good belief states may have formed earlier than T and good behavior should be strengthened. Similar the model should not strengthen belief states that lead to incorrect class labels. This is different from the original RAM implementation where all states at $t = 1, \dots t = T$ are strengthened when the action network outputs a correct class label at time T.

Given that we are implementing a belief-MDP and the corenet provides us with the belief state $b(s)$, the action network should approximate $p(y \mid b(s), x)$. This is a probability distribution over all class labels y given the current belief state and the image x for which that belief state has been formed. To transform the output of the action network to a probability distribution a Softmax function is used to normalize the outputted values so that they sum up to 1. The action network then resembles a categorical distribution where the number of categories equals the number of class labels.

As shown in equation 4.1 reward can be defined as the outcome of the likelihood function when the predicted class label $\hat{y}$ equals the actual class label y

$$r_t = \begin{cases} \text{argmax}_{\hat{y}}\, p(\hat{y}_t \mid b(s)_t, x), & \text{if } \hat{y}_t = y \\ 0, & \text{if } \hat{y}_t \neq y \end{cases} \tag{4.1}$$

Ba et al. (2014) however recommend using a 0/1 discrete indicator to reduce variance, as shown in equation 4.2

$$R = \begin{cases} 1 & y = \text{argmax}_{\hat{y}}\, p(\hat{y} \mid b(s), x) \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Using a Monte-Carlo procedure we collect all rewards starting from $t = 1$ until we reach the maximum number of glimpses at T. This sequence of rewards will then form the estimate of the total expected return for the whole episode.

$$G = \sum_{t=1}^{T} \gamma^t r_t \tag{4.3}$$

The collected rewards can then be used by the Reinforcement Learning algorithm to optimize the current policy such that the total expected return will be maximized. This procedure will only work if the action network itself also behaves in an optimal manner. Since all individual components in the RAM model start from scratch, this has to be trained together with the policy network. As mentioned in the previous chapter, there is a reciprocal relationship between inferring and acting, meaning that the model can only work well when they are both optimized.

For the classification tasks used in this research this implies that the policy network can only learn from rewards when the action network outputs the correct likelihood function. On the other hand the action network can only learn to predict the correct class labels when the policy network provides the correct sampling locations. Thus the model must in fact optimize two loss functions, one for the

policy and one for the action network. In the case of the action network this can be done supervised by optimizing a cross-entropy loss function. This is given by:

$$H(p,q) = -\sum_i p_i log q_i = -y \log \hat{y} - (1-y)\log(1-\hat{y}) \tag{4.4}$$

Where p is the true distribution and q the approximation of p

Policy network

Similar to the action network, at each t, the policy network will receive the updated belief state from the core network. Using a single fully connected layer it will process this belief state to determine the next location to extract the image patches. These locations are sampled from a Gaussian distribution for which the policy network will provide the means. The standard deviation is fixed and therefore a hyperparameter.

Because we are dealing with two dimensional images, the network will output two mean values. Although the image surface falls in the range $(-1, 1)$, the outputted mean values are not restricted and are theoretically in the range $[-\infty, \infty]$. Restricting either the means or the sampled locations from the Gaussian distribution to fall in the range $(-1, 1)$ was tried, but only increased the risk of ending up with an unstable network. When the policy network samples partially or completely outside the image, it will receive zero values for all pixels not in the image range. Thus the input image is also of infinite size and the policy network should learn to only output values which are within the correct range, something which in practice did not seem to be a problem as long the initial location coordinate values l_0 are between -1 and 1.

As mentioned when discussing the action network, we need to optimize two loss functions. For the policy network an advantage actor critic method will be used, similar to the ones described in equations 3.33 and 3.36. There are however some implementation differences which should be mentioned. First the reward collected at $t = 0$ is not used, because this is not the result of the policy network, but was randomly sampled from a uniform distribution. Second, since we always sample full episodes we also

use the reward collected at $t = T$, which in turn implies that we should not replace our final reward with the output of the value function. Our reward function then simply becomes the sum of discounted rewards.

$$\hat{R}_t = r_t + \gamma_{r_t+1} \dots \gamma^{T-1} r_{T-1} \tag{4.5}$$

The baseline used to reduce variance is the State-Value function. For the RAM model this has the consequence that we now need an extra neural network which is capable of estimating the true State-Value function and an extra loss function to optimize. Because we are dealing with a belief-MDP these states are the belief states outputted by the core network. This means that the loss function we need to optimize the state-value function changes to:

$$\| R(b(s)_t) - V^{\pi_\theta}(b(s)_t) \|_2^2 \tag{4.6}$$

Because all parameters are shared in the RAM model, we end up optimizing a loss function which is the sum of the cross-entropy loss, the policy gradient loss function and the value loss.

4.1.2 Deep Recurrent Attention Model

The basis for the Deep Recurrent Attention Model (DRAM) is the RAM model described above. The biggest differences are the addition of an extra recurrent layer and the use of a context network (Ba et al., 2014). See figure 4.2 for the schematic diagram.

Extra recurrent layer

In the RAM model, one recurrent layer encodes all the information needed for both the action network as well as the policy network. In the DRAM model an extra recurrent layer is added on top of the first. This is done for two reasons. First it enables the model to form a separate representation for the policy. Second it allows the use of a context layer. Since the context layers uses a lower resolution version of

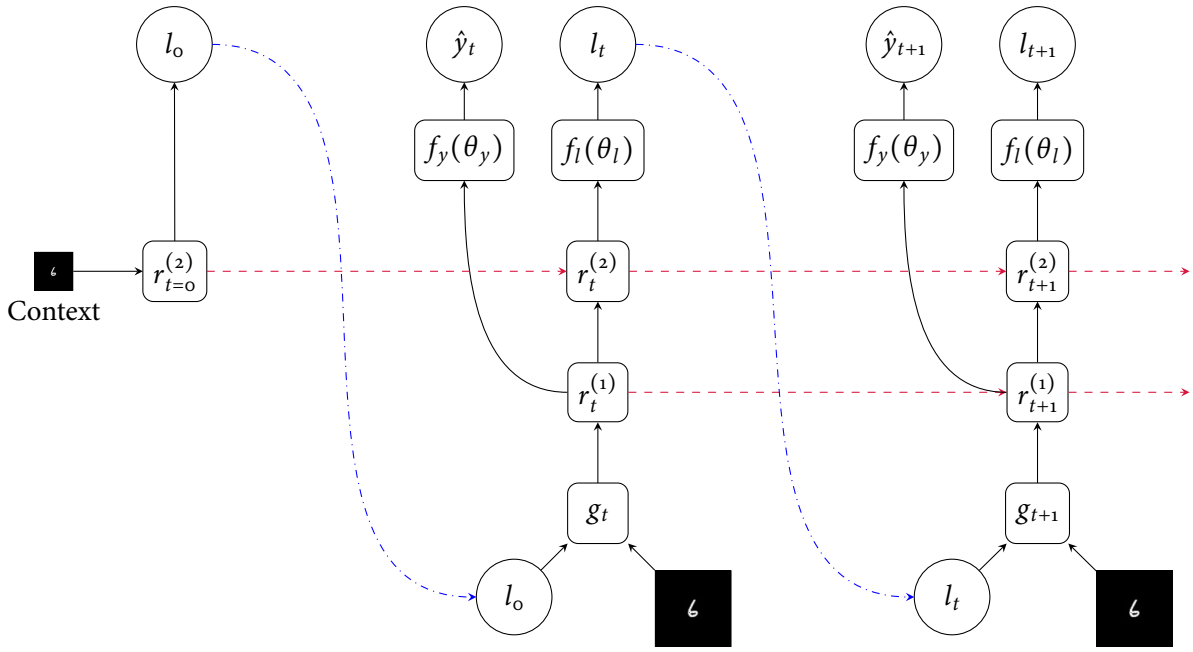


Figure 4.2. Deep Recurrent Attention Model. Below is an explanation of the components that are added in comparison to the RAM model. See figure 4.1 for the explanation of those elements.

Context = a lower-resolution version of the complete image is presented to the context layer and outputted as initial state for the second recurrent layer $r^{(2)}_t$

$r^{(2)}_t$ = extra recurrent layer which receives input from $r^{(1)}_t$ and is initialized with the context state at $t = 0$.

l_0 = initial sampling location is now determined by sending the initial context state to the location network.

the full image, adding this to the first recurrent layer would give the action network a shortcut to that image. You then run the risk that the model can learn faster from the context image than the image patches and you end up with a local optimum which is far from the actual optimum.

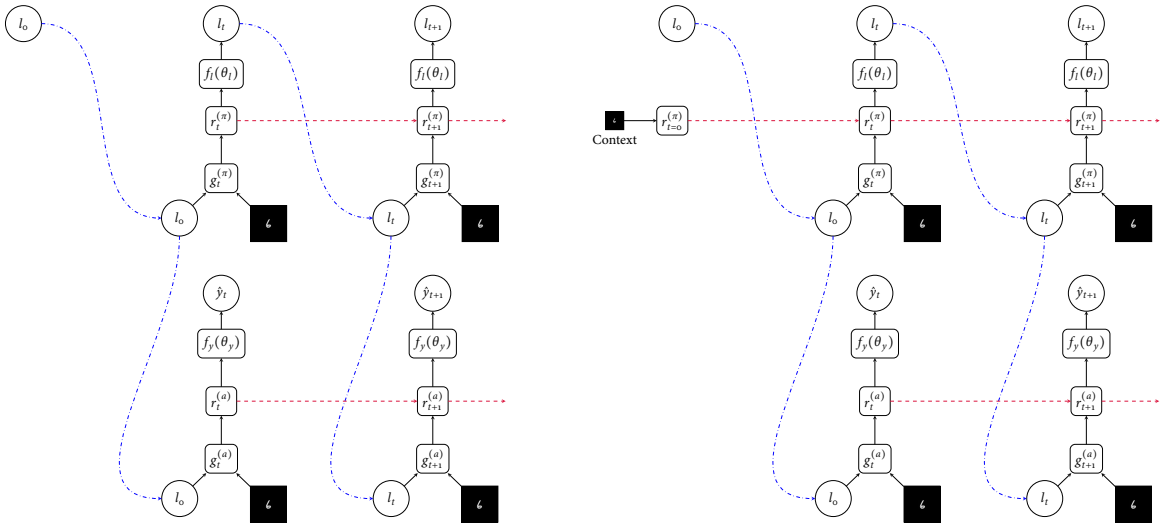
Input to the first recurrent layer is the same as in the RAM model, however it's output is now send to the action network and the second recurrent layer.

Context

A down scaled version of the full-image is presented to a fully connected layer. It's output is used as the initial state of $r^{(2)}$ and as input to the policy network to get the initial sampling coordinates. This should give the model a more informed initial glimpse location.

Besides that having the initial state set to the context allows the recurrent network to integrate this information with the glimpse information from the previous recurrent layer $r^{(1)}$. This should

enable the policy network to create a representation of the layout of whole image. Something which is useful for predicting where the most interesting regions of an image are located and should lead to a more accurate policy and minimize the need for extra zoom levels. It basically replaces the peripheral information coming from the extra zoom levels in the RAM model. As an extra benefit this means that the action network is now only trained using the smaller high resolution patches. This should reduce noise in the input signal.



(a) Separated Recurrent Attention Model

(b) Separated Recurrent Attention Model with Context

Figure 4.3. Two models with a separate action and policy network. Models are similar with the exception of having a context (b) or not (a)

$g_t^{(\pi)}$ = policy glimpse network.
$g_t^{(a)}$ = action glimpse network.
$r_t^{(\pi)}$ = recurrent layer forming the belief state for the policy network
$r_t^{(a)}$ = recurrent layer forming the belief state for the action network
$r_t^{(2)}$ = extra recurrent layer which receives input from $r_t^{(1)}$ and is initialized with the context state at $t = 0$.
l_0 = initial sampling location is now determined by sending the initial context state to the location network.

4.1.3 Models with separate parameters

Inspired by the RAM and DRAM models, two new models were developed with separated action and policy networks. The first, using the RAM model as base, has two networks with a single recurrent layer and will send it's output either to the action $r_t^{(a)}$ or policy network $r_t^{(\pi)}$. The second has the same

action and policy network structures as the first, but it now uses, similar to the DRAM model, a context network to initialize the initial state of the recurrent layer of the policy network $r_t^{(\pi)}$. One difference however is that the initial sampling location is the result of sampling from a random uniform location for both networks. The context state is not used for determining that location, since preliminary results did not show any advantage of doing that over random sampling.

Both models described above have shared parameters, which seems to be the most obvious choice when implementing active sensing network this way. This is because the input image is the same for both the action and policy network and we have a reciprocal relationship between inferring and acting. However there is nothing theoretically that prevents using separate action and policy networks.

There are several advantages from having separate networks. For instance the action network can be presented with a smaller high resolution image patch, while the policy network works with a larger, but lower resolution patch. It thus allows for more flexibility in what information is presented to both networks.

There are more benefits, first having a separate policy and action network will lead to completely specialized networks. It can be argued that the information that is needed to determine the correct class label is not necessarily the same as what is needed for determining the next sampling location. By optimizing $r^{(1)}$ in both the RAM and DRAM model to form representations for both networks might then even lead to a belief state which is sub-optimal for both. By learning separate belief states they might become more specialized and lead to better representations for the following network.

Another benefit is from using separate optimizers. In the RAM and DRAM model all the losses are summed to form a overall loss which is then minimized by a single optimizer. This also means that you can just specify one learning rate for all the losses. With multiple optimizer you are not restricted to using the same optimizer and learning rate for all losses.

4.2 Improving the base models

First is the learning algorithm. These models use the Reinforce algorithm, which is not optimal since it suffers from high variance and can lead to reaching a local optimum which is too far off the optimal policy. Different other training algorithms exists that should give better performance than Reinforce. In this thesis I will compare the models trained with Reinforce against an algorithm called Proximal Policy Optimization (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017).

Second is the baseline technique used. According to Tucker et al. (2018) a different baseline technique called the Horizon-Aware value function should perform better than the normally used state-value function.

First the PPO algorithm will be described in more details, followed by explaining the horizon-aware advantage function. After that the structure of the models will be discussed followed by a description of the different experiments.

4.2.1 PPO

The training method that will be used in this experiment is called the Proximal Policy Optimization developed by (Schulman et al., 2017). This algorithm was developed because of the success of a previous algorithm called the Trust Region Policy Optimization (TRPO). (Schulman, Levine, Abbeel, Jordan, & Moritz, 2015). The success of is TRPO is because it defines a so-called trust region.

A trust region is a region in which the local approximations of the function are accurate. TRPO ensures that the updated policy does not deviate to much from the previous policy. To do so a surrogate objective is maximized.

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \mathbb{E}\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] \\ \text{subject to} \quad & \mathbb{E}[KL[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta \end{aligned} \tag{4.7}$$

With θ_{old} vector of parameters before the update. The parameter δ controls the maximum kullback leibler divergence between the old and the new policy. The problem can then be solved by using a conjugate gradient method. The latter is also a downside of TRPO, since it makes it more complicated and less flexible than algorithms using a gradient descent procedure. PPO was developed to have the same advantage of having a trust region, but with a much simpler implementation. There are multiple variants of the PPO algorithm, but the one used in this thesis is clipped PPO.

The objective proposed for PPO is

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{4.8}$$

With ϵ being the clipping parameter.This new objective implements a way to do a trust region and is compatible with stochastic gradient descent. It can be implemented as an easy modification of an existing policy gradient method.

4.2.2 Baseline

As mentioned in section 3.3.2 variance in the policy gradient estimation can be reduced by fitting a baseline function. The most optimal baseline would be the true state-action value function, but the most commonly used one is the state-value function as an unbiased estimate of the Q-function, giving an advantage function.

According to Schulman, Moritz, et al. (2015) the Q-function might be optimal in theory, but in practice it is much more difficult to train than the state-value function. Tucker et al. (2018) evaluated different baseline variance reduction techniques. They came to the conclusion that state-action dependent baselines were unable to achieve significant variance reduction and that much larger gains could be achieved by improving the accuracy of the state-value approximation. Motivated by this result the authors came up with a new approach for approximating the state-value function, called the horizon-aware value function.

This approach features a neural network that outputs two state values, which are combined with discounted time to form a value function estimate. Motivation for this approach is that standard implementations for the value function do not take into account that the expected return near the end of an episode will necessarily be small.

$$\hat{V}(s_t) = \left(\sum_{i=t}^{T} \gamma^i\right) \hat{V}_1(s_t) + \hat{V}_2(s_t) \tag{4.9}$$

With $\hat{V}_1(s_t)$ being the expected discounted return averaged over time and $\hat{V}_2(s_t)$ a state-dependant offset. Tucker et al. (2018) show that this approach outperforms other state-of-the-art methods. In this theses both the standard state-value advantage function will be used as well as the horizon-aware value function.

4.2.3 Exploration noise

Two methods of improving exploration by adding noise to the model parameters are tested. In the standard situation locations are sampled from a Gaussian distribution with a fixed standard deviation and exploration is only possible because of the stochasticity introduced by using such a probability distribution . Two methods which add noise to the network weights should give a wider range of possible selected location and should therefore in theory lead to better exploration. In the experiments this noise is only added to the parameters of the fully-connected layer of the policy network, which outputs the mean of the Gaussian distribution.

The first of this methods developed by Fortunato et al. (2017) is implemented by transforming a normal fully-connected layer to a, what the authors call a noise-layer in which weights are perturbed by noise. The amount of noise added to the weights is done by a parametric function whose parameters are trained using a gradient descent procedure.

The second method is based on the procedure described by Plappert et al. (2017). In this method a copy of the existing model is made and noise is added to the weights in this copy. Actions are then

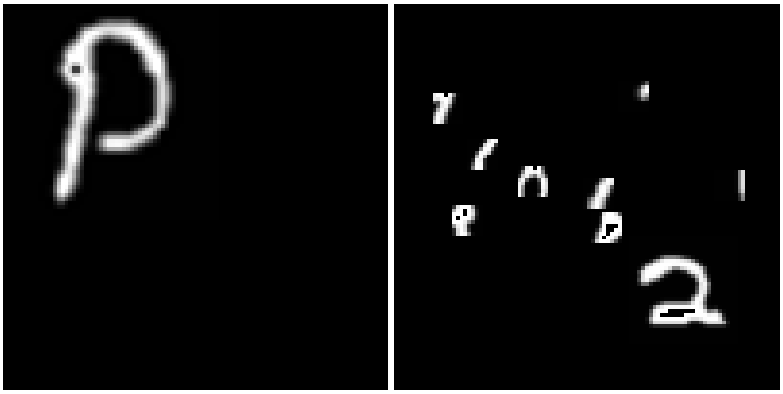
selected from this copied network and are used for training the original model. To prevent that the copied policy deviates too much from the original the amount of noise added to the copy is regulated by a distance parameter. In this setting the average euclidean distance between the mean coordinates selected from the normal and the copied policy.

4.3 Tasks

The models will be trained on four different tasks, which are inspired by the tasks used in the RAM and DRAM papers, but should be more challenging and cover more aspects of visual attention. The tasks show an increase in complexity and are performed in that order. The first two tasks, which are called the ”Translated EMNIST” and the ”Cluttered EMNIST” are used to test all four model architectures with all possible combinations of training algorithms, value functions, and parameter space noise. This leads to a total of 48 experiments per task.

From the outcome of those tasks a selection will be made out of the best performing configuration to test them against two even more challenging tasks, called the ”Cluttered MNIST Sum task” and the ”Cluttered MNIST Mixed Sum Task”

4.3.1 Translated EMNIST and Cluttered EMNIST



(a) translated EMNIST task. *(b)* cluttered EMNIST task.

Figure 4.4. Examples from the translated (left) and cluttered EMNIST tasks

Both the translated and cluttered EMNIST are based on their MNIST counterparts in the RAM and DRAM papers. The MNIST dataset, a collection of handwritten digits from 0 to 9, is one the most widely used sets for training classifiers, but has been under fire in recent years for not being challenging enough. Highest accuracies are already above 99.7%, which is the main reason for Cohen, Afshar, Tapson, and van Schaik (2017) to create a more challenging alternative called the Extended MNIST (EMNIST).

The MNIST dataset is actually derived from a larger dataset called the NIST dataset, which besides numerical digits contains handwritten uppercase and lowercase letters. Using a similar procedure than the one used for for creating the MNIST dataset a new dataset was created containing both numbers and letters. The full version of this Extended MNIST dataset now contains 62 classes. However some classes are too difficult too distinguish, therefore the authors have created a more balanced version of the EMNIST dataset containing 47 classes. The number of digits remained the same, with 10 classes, but the number of letter classes was reduced from 52 to 37. This balanced dataset is used to create the stimuli for the ”Translated” and ”Cluttered” EMNIST tasks.

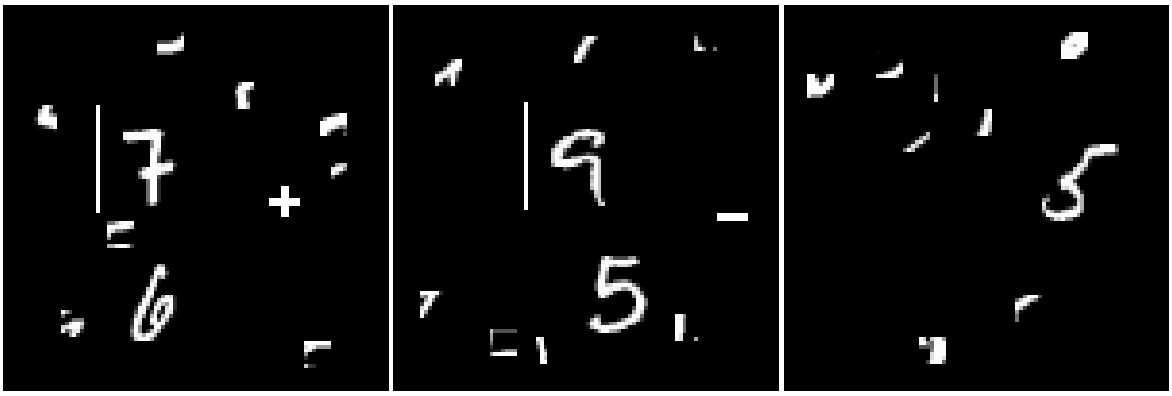
Figure 4.4 shows an example from both tasks. In the ”Translated EMNIST” a randomly selected character, 28x28 pixels, is placed at a random location on a 100x100 pixels black image. The obvious challenge with this task is that the policy first has to determine the location of the character before using that same policy to focus on those parts of the characters that provide the most useful information for the action network to determine the correct class.

The ”Cluttered EMNIST” task is an extension of the translated version. In a similar manner a character is placed randomly on a black image, but now 8 pieces of ”clutter” are also placed on that black image. Those pieces of clutter are in fact extracted from the same EMNIST dataset. From 8 randomly chosen characters, a rectangle of 8x8 pixels is extracted. The location where that part is extracted is also randomly selected.

The added challenge now is that the because the clutter is extracted from the same dataset as the target character, these pieces of clutter have shapes that may also be present in the targets. This should

make it more difficult for the policy to learn to ignore those distractors and focus on the target character instead.

4.3.2 Cluttered MNIST and Mixed MNIST Sum tasks



(a) *(b)* *(c)*

Figure 4.5. Figures a and b show plus and minus variants of the cluttered MNIST sum tasks. The Mixed Sum task is a combination of that same SUM task and cluttered MNIST digits.

In the previous two tasks the challenge was mainly to find the target character, either with or without distractors. In the MNIST and Mixed MNIST sum tasks the difficulty is increased by forcing the model to find one or more randomly placed targets among distractors and integrate the information in a correct way. Note that for this task stimuli are created using the MNIST dataset instead of the EMNIST dataset used for the tasks mentioned above because now only numbers are of interest.

The first task that tries to accomplish this is the ”Cluttered MNIST Sum” task. In this task MNIST digits are combined with either a plus or minus operator to form addition and subtraction sums. Two digits are used with one operator, which results in answers that range from -9 to +18. This is then transformed in a classification problem with 28 possible classes. The three target characters are randomly placed on a black background of 100x100 pixels and by using a similar procedure as the previous cluttered task, 8 pieces of clutter are randomly placed on that background as well.

There is a problem with using the minus operator and that is that the order of the digits matters. The answer to 6-4 is different than the answer to 4-6. To solve this, the first digit in order is marked with a vertical bar on the left. This increases the difficulty of the task, because now the model not only needs

to find the randomly placed targets, but has to figure out what the vertical bar means in conjunction with the operator sign.

The second task, the Mixed Sum Task is an extension of the Cluttered Sum task because it is largely the same task but now the sums are mixed with stimuli containing randomly placed MNIST digits and random clutter. The reason this task was created was to investigate whether the active sensing models are able to discriminate between a classification and a sum task. For good performance knowledge about both tasks is necessary and context should provide the information about which task is at hand.

Figure 4.5 gives a good example from which stimuli both tasks are made up. The first two images show examples of a plus and minus sum and the third is an example of a Cluttered MNIST which is added to the Mixed Sum task.

4.4 Implementation Details

In this section the implementation details of the models will be discussed. First the details that are the same for all experiments, followed by the differences between the experiments.

All models were build using Tensorflow 1.12. (Abadi et al., 2015). Experiments were performed on a computer featuring an eight core Intel Xeon processer at 2.4ghz, 64GB RAM and two Nvidia Tesla K80 GPUs.

All the hidden layers in the four described models use ReLU activation functions, $f(x) = \max(0, x)$. The recurrent layers use Long Short-term Memory (LSTM) celss, with the exception of the action network for the separated models which uses Gated Recurrent Units or GRU cells(Cho et al., 2014; Hochreiter & Schmidhuber, 1997). The policy network uses either a dense layer to output the mean without a non-linearity applied, or is the output of a noise layer when that exploration noise option is used. The standard deviation for the Gaussian Policy is not trained, but set to a fixed value of 0.08.

For the training procedure batch sizes of 128 were used and every experiment was trained for 30.000 epochs. Rewards were discounted using a factor of 0.95. The optimizer used is the Adam optimizer (Kingma & Ba, 2014)

Stability was an issue for the recurrent networks because of exploding gradients. To prevent this from happening gradient norm clipping was applied on the gradients before optimizing. The clipping value was estimated empirically by observing the development of the average norm during training and setting it to the largest value possible (Pascanu, Mikolov, & Bengio, 2012).

Translated EMNIST and Cluttered EMNIST

For both the Translated and Cluttered EMNIST tasks, a glimpse window size of 12x12 pixels was used. A zoom level was added, meaning that a patch of 24x24 pixels was also extracted, but is downsized to 12x12 pixels to reduce it's resolution. All the layers in the models have 128 units. The glimpse network is processing it's visual input using a single dense layer and the location is processed by a single dense layer as well. Similar to the glimpse network, the context is processed by a single dense layer. The number of glimpses is set to 6.

Somewhat problematic was the way the Tensorflow function to extract image glimpses works. When focus is completely or partially outside the stimulus image, the part not belonging to the image is replaced by noise. It was found that this noise had a severe negative effect on the training procedure for models using a glimpse network with a single dense layer. Because of this many models became unstable as soon as they started focusing on the edges of the stimulus image. To prevent this, a mask containing zeros was laid over the glimpse window to remove the noise. Since there are no boundaries restricting the focus locations, it means that the stimulus images are surrounded by infinity zeros.

Cluttered MNIST and Mixed MNIST Sum tasks

For both sum tasks 8 glimpses are allowed and the glimpse window size is set to 24x24 pixels, with a single zoom level of 48x48 pixels. This zoom level is then downsized to 24x24 pixels. With the larger window sizes, using convolutional layers has become a serious option. For the smaller glimpse size of 12x12 pixel, using convolutional layers did not seem to be of any benefit and was merely causing longer computation time.

The glimpse network for the sum tasks was given two convolutional layers, each with 12 filters. The first layer with a kernel size of 5x5 followed by a kernel size of 3x3 for the second layer. The output of convolutional layer 2 is then further processed by a dense layer containing 256 units. No pooling or dropout was applied. Masking the noise was also not necessary for models using convolutional layers and was applied.

The glimpse location is processed by the glimpse network using a single dense layer with 256 units.

Using a similar setup as the glimpse network, the context was also processed by two convolutional layers and a dense layer. All the other hidden and recurrent layers have the same number of 256 units.

5 Results

In this chapter the performance of the active sensing models on the four tasks will be discussed. The first section will talk about the effect the different adjustments made to the RAM and DRAM have on task performance. Based on those results a selection of eight models was made to further investigate the policies that were learned for the given tasks.

5.1 Testing Model Configurations

Before investigating whether or not artificial active sensing models are capable of learning an active sensing policy, all the 48 possible combinations of shared vs separate parameters, exploration noise, advantage function, and having an extra context layer or not are investigated using the Translated and Cluttered EMNIST tasks. Goal is to estimate the effect of the different configuration options so that a good good selection of models can be made for assessing the learned policies.

Two types of data were collected for this purpose. First raw classification scores were gathered using validation datasets containing 100000 images in the same order for each of the 48 configurations. From this data mean validation scores for each possible configuration were calculated. For the Translated EMNIST task these scores can be seen in the table appendix A.1 and table B.1 has the validation score for the Cluttered EMNIST task. The same raw classification scores were then used for a bootstrap procedure to estimate the effect for the five configuration options.

Second, during the training procedure a separate dataset was used to periodically calculate an average mean classification score. This information will be used to investigate the development of that mean

validation score over time. Ideally this score will increase over time before reaching convergence, a point in time after which the score will not further improve. Since there seems to be much difference between all the mean validation scores on both experiments, it will be interesting to see what the effect of the different configurations is on the training process.

5.1.1 Estimating configuration effects

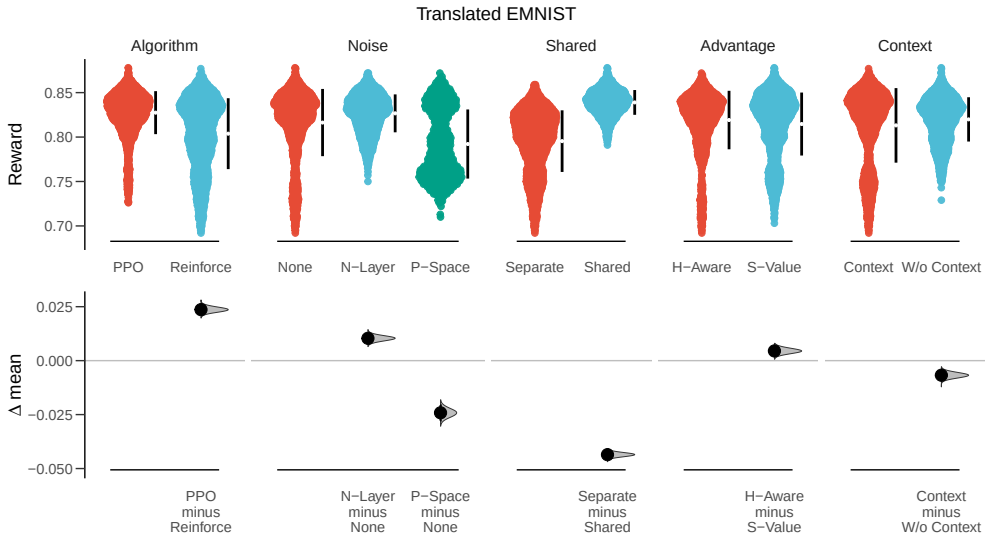
To get a good estimate of possible differences between the validation scores within each configuration option, estimation plots using the procedure described by Ho, Tumkaya, Aryal, Choi, and Claridge-Chang (2019) were created. The resulting plots are based on the collected raw data, but since they contain 100.000 validation samples for each of the 48 possible configurations the combined dataset was too large for the required bootstrap procedure. To solve this, instead of using less samples, the total number of data points was reduced by transforming the 100.000 validation samples to 100 equal sized bins containing the means of 1000 validation samples. After this reduction the dataset now only contains a total of 4800 data points instead of the original 4.8 million.

A second problem is that when looking at all the results in tables A.1 and B.1, it is clear that a fairly large proportion of the trained configurations either completely failed to converge or ended up with a low mean classification score. Because the intention is to find configuration options that lead to better models, it was decided to leave the bad performing models out of the bootstrap procedure.

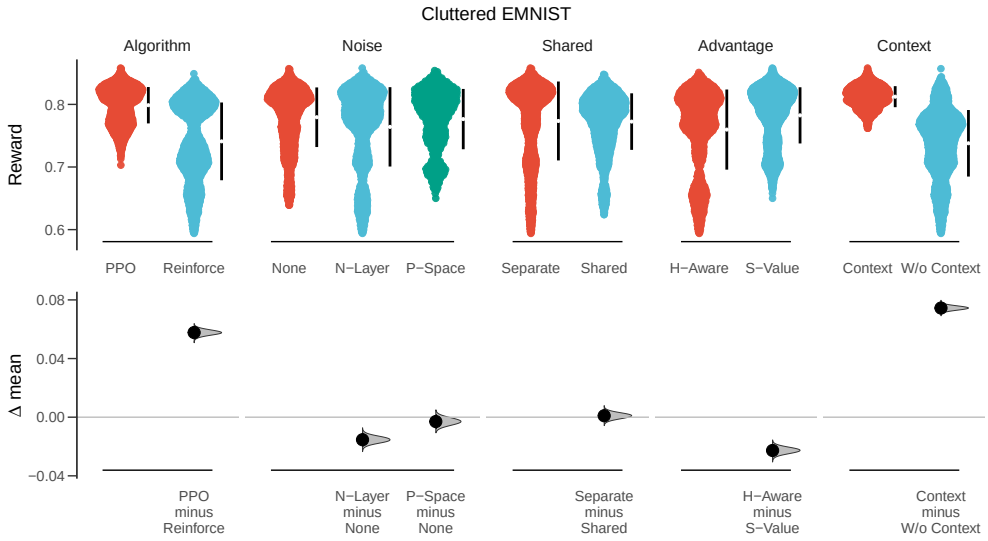
Instead of subjectively choosing a cut-off point, k-means clustering was performed on the 4800 binned classification scores. By repeating this procedure for $k = 1 \ldots k = 10$ it became clear that for both experiments the optimal number of clusters is two. By labeling these clusters as ”good” versus ”bad”, the cut-off point is then set at the highest mean validation reward value in the ”bad” cluster. For the Translated EMNIST task this value is 0.457 and 0.375 for the Cluttered EMNIST.

It is not a good idea to use the cut-off point to exclude individual data points, since excluding them introduces a bias towards a too positive overall score. Instead all configurations that contain a least one overall mean validation score lower than the cut-off point were excluded for estimation. By applying

this procedure, 11 configurations were excluded for the translated and 12 were excluded for the Cluttered EMNIST task.



(a) Translated EMNIST



(b) Cluttered EMNIST

Figure 5.1. Estimation plots for the translated (a) and cluttered EMNIST (b) tasks. *Top half*: Swarm plots displaying the distribution of mean validation scores grouped per configuration option. Vertical black lines give the mean ± 1 standard deviation. *Bottom half*: Differences in mean reward scores within each group. Grey area right of each Δ mean estimate shows its 95% confidence interval.

The resulting estimation plots for both experiments can be seen in figure 5.1. A nice thing about these estimation plots is that they summarize many important aspects of the data. The swarm plots on the top half, combined with the black vertical lines (mean ± 1SD) give a good indication of the distribution of the mean validation reward scores. The mean difference estimations on the lower half are combined

with confidence intervals to provide a clear visual method to inspect the differences within each of the configuration options.

For the Translated EMNIST task, see figure 5.1a, differences are relatively low. The y-axis scale for the Δ mean plot reveals that the group differences all range between 0.025 and -0.05, with the difference between shared and separate configurations being the largest. Models with shared parameters are performing better than models with a separate action and policy network. Not only is the mean validation score higher for the shared parameter models, variance also seems to be much smaller in this group. For the learning algorithm, it seems that the PPO algorithm has a small advantage over the Reinforce algorithm. When looking at the remaining options, there does not seem to be a real benefit of using exploration noise, the difference between the two advantage functions is close to zero, which is also the case for having a context layer.

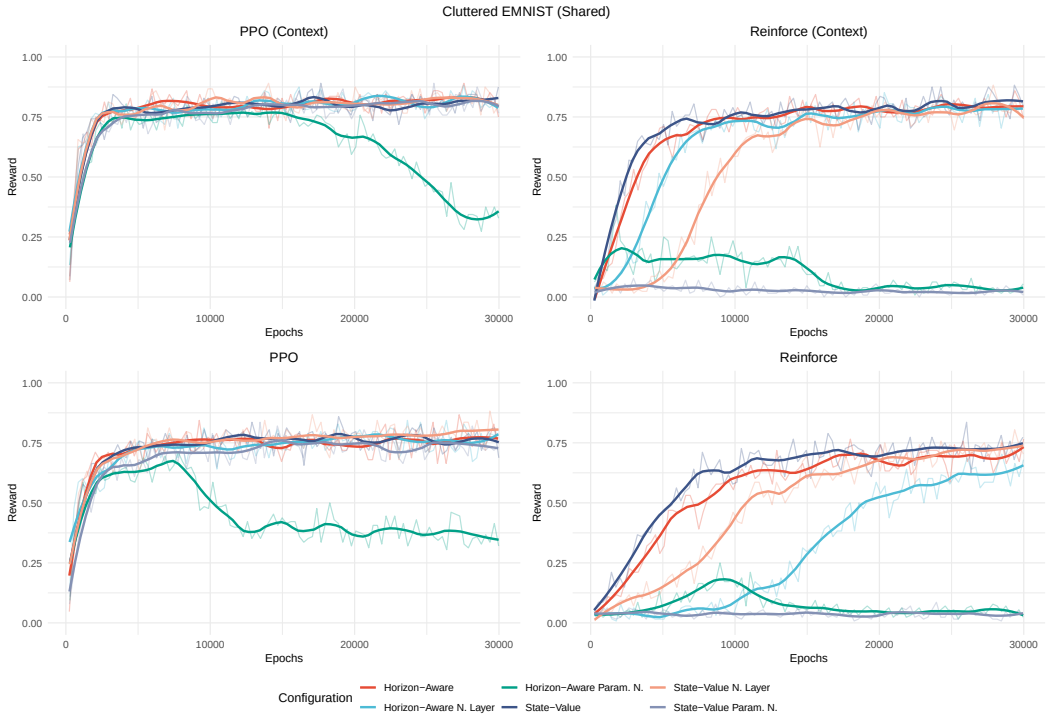
With the more complicated Cluttered EMNIST task, see figure 5.1b, differences within configuration groups become more apparent. Having a context layer seems to be clearly beneficial as is using the PPO algorithm. This is in line with table B.1 in which out of the 10 highest scoring configurations 9 use a context layer and also 9 use the PPO algorithm.

5.1.2 Training progress

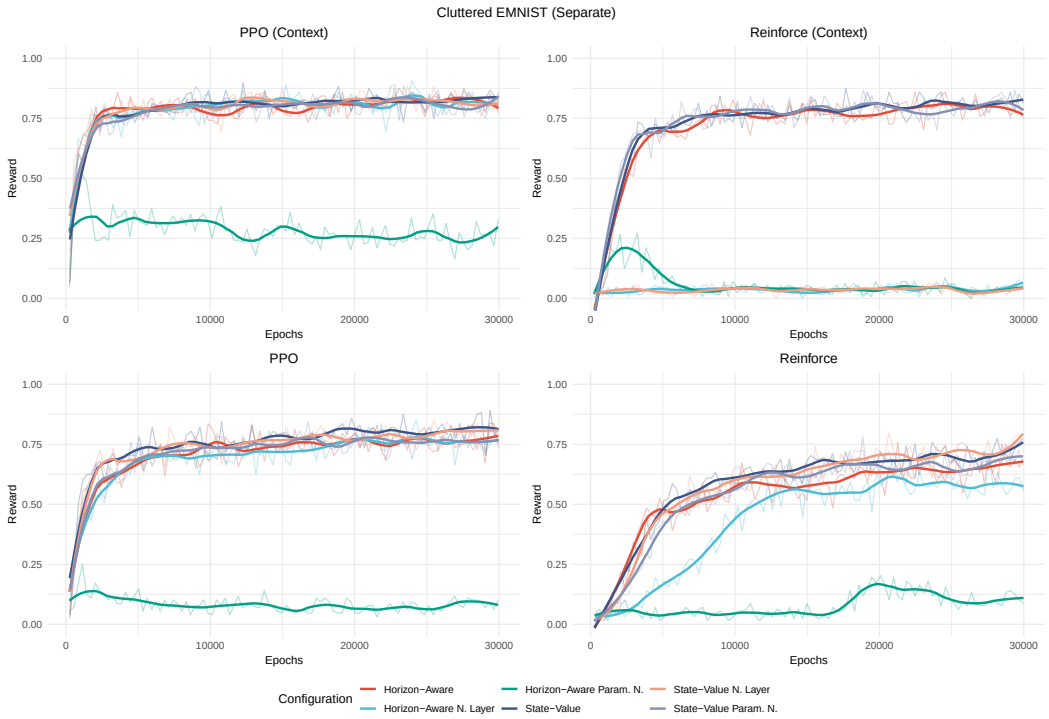
To gain more insight in the differences between the configuration options, plots displaying the development of the mean validation reward score during training were also produced. The mean reward value was calculated on a regular interval during training on a batch of 128 validation samples.

Because the training progress for the Translated EMNIST task was found to be less informative than the training progress on the Cluttered EMNIST task, it's training plots are placed in appendix C.1. The training plots for the Cluttered EMNIST task can be seen in figure 5.2.

What strikes, when looking at the training plots, is the difference between PPO and Reinforce in the time needed for converging. Except for combinations utilizing PPO, Horizon Aware, and Parameter space noise, all of the PPO configurations seem to converge before 10000 epochs, with



(a) Shared parameters



(b) Separate parameters

Figure 5.2. Development of validation reward during training on the Cluttered EMNIST task for all 48 configurations. Split on shared vs Separate Parameters and further divided by algorithm and context. Lines display the baseline used and the type of exploration noise (Parameter space, Noise layer or None).

several even reaching their maximum before 5000 epochs. In contrast many configurations using Reinforce, especially those without a context need more than 10000 epochs to converge. It can also be seen that most of the not converging or ”bad” cluster models use the Reinforce algorithm. The learning algorithm however does not seem to be the biggest cause for bad performance, since all of the configurations belonging to the ”bad” cluster for both tasks use one of the two exploration noise options with parameter-space noise being the most dominant.

The training graphs reveal another important point about the parameter-space noise option. All of the configurations that are in the ”bad” cluster, but still seem to learn something use that type of exploration noise. Some of those configurations actually started with performance similar to the ones in the ”good” cluster, but after some time training progresses performance collapses. This is an indication that this type of noise can introduce instability. It must be noted however that for configurations using the PPO algorithm, only combinations with parameter-space noise and the horizon-aware baseline seem to be failing. This in contrast to Reinforce, for which also combinations using the state-value baseline and the noise-layer option may fail.

The benefits of having a context is harder to explain. The Cluttered EMNIST training graphs in figure 5.2 point in the direction that having a context has a stabilizing effect on training configurations using the Reinforce algorithm, although the same cannot be said when looking at the progress on the translated EMNIST task, see figure C.1.

As the estimation plots in figure 5.1 already indicate, for the Translated EMNIST task having a context does not have much of an effect on classification performance. Even when split on learning algorithm, overall absolute mean differences between having a context or not is less than 0.001. Things are different for the Cluttered EMNIST task. As expected the largest difference between having a context or not is when the Reinforce algorithm was used. Reinforce with context has an average classification score of 0.8, while Reinforce without a context only has an average of 0.69. Differences for the PPO algorithm are smaller, but still clearly present. PPO with context manages to get an average classification score of 0.82, whereas the configurations without context have an average score of 0.79.

Possible explanations for the observation that having context is more beneficial for the Cluttered EMNIST task are the fact that the target character used is smaller (28x28 pixels) than in the Translated EMNIST (56x56 pixels) task and the presence of distractors. The context layer, which uses a low-resolution version of the whole image, may assist by identifying valuable locations on forehand that can be combined with the high resolution data received from the glimpses. For the Translated EMNIST task, finding the character is less of an issue and determining next glimpse location based on the previous glimpse may be more important.

The reason that the helping effect of a context layer seems stronger for the Reinforce algorithm may have the same origin as to why exploration noise configurations fail more often in combination with that algorithm. PPO seems to be more resilient against disturbing factors like distractors or noise. This may also explain why it converges faster than Reinforce. PPO is better capable of extracting useful information from a noisy environment. If this is true, then the differences between the algorithms should increase when tasks become more complex.

5.2 Active sensing policy

The main purpose of this research is to investigate whether artificial neural networks are capable of learning an active sensing strategy. Following Gottlieb (2018), key aspects are dependence on prior knowledge of the task structure, the expected information gains associated with sampling a cue, and the fact that the decision to sample at a certain location is made before discriminating.

Analyzing the learned policy for all the ”good” configurations on the different tasks would not be useful. It was decided to create a selection of eight configurations using the results described above. The resulting configurations can be seen in table 5.1. From this point on, all analysis and follow-up experiments will be done using that same selection of eight configurations.

Table 5.1

Eight selected model configurations

Model	Algorithm	Context	Noise	Shared	Advantage
A	PPO	Yes			Horizon-Aware
B	PPO	Yes			State-Value
C	PPO	Yes	Noise-Layer	Yes	State-Value
D	PPO			Yes	Horizon-Aware
E	PPO				State-Value
F	PPO	Yes	Noise-Layer		State-Value
G	PPO	Yes		Yes	State-Value
H	Reinforce	Yes		Yes	State-Value

5.2.1 Task Performance

The first requirement for an active sensing policy, is that it has good prior knowledge of the task. This requirement can be translated into showing good performance on the required task. This can be justified, because reaching good performance on all the tasks should only possible when the models have good knowledge about what targets they should find and how they should use the acquired information to identity the correct class label.

Task performance is measured using the mean validation scores. First task performance on the Translated and Cluttered EMNIST tasks will be discussed, for which the overall validation mean scores were already mentioned in the previous section and are depicted in tables A.1 and B.1. Second task performance on the Cluttered MNIST Sum and Mixed MNIST Sum tasks will be assessed for the eight models.

Translated & Cluttered EMNIST Tasks

Both the translated and cluttered EMNIST task can be seen as consisting of two sub tasks that need to be performed in order. First locating the target character and second identifying the target character. Finding the target is a challenge in itself, with just 6 glimpses the maximum percentage of the surface that can be seen is only 8.6% in high resolution or 34.6% in lower resolution. The target EMNIST characters are either left to their original size (28x28 pixels) in the Cluttered EMNIST Task or enlarged with a factor 2, to 56x26 pixels for the Translated EMNIST task. Given that the pixel distribution for all

the EMNIST characters seems to resemble a Normal distribution, see figure D.1, it can be determined that the pixel coverage of an EMNIST character is less than 50% for more than 98% of all characters. This means that for the Translated EMNIST task, less than 31% will be covered by that actual character, a value that decreases to just 4% for the Translated EMNIST task.

To have some reference for determining how good these models are in performing the tasks, it is useful to know what the current state-of-the art classification scores are on the balanced EMNIST task. Only the actual classifying part of the task can be compared this way, but since classifying the target without locating it would be impossible it gives a good indication of how the different trained models perform.

The original creators of the EMNIST dataset report a classification score of 78.02% ± 0.92%, while at the time of writing the highest mentioned score in literature is 90.46% ± 0.22% and the second highest mentioned is 88.3% ± 0.8% (Cohen et al., 2017; Dufourq & Bassett, 2017; Jayasundara et al., 2019). It must be noted that those scores were reached using the 28x28 pixel images from original balanced EMNIST dataset.

Given that the models are trained on more complicated stimuli and were never optimized for reaching reaching state-of-the-art classification scores, it is good to see that many of them have scores that are higher than 78%. In fact the scores reached by the eight selected configurations range from 81.4% to 85.1% for the translated and 77.9% to 82.8% for the cluttered EMNIST task, which is still pretty close to the state-of-the-art.

Cluttered & Mixed MNIST Sum Tasks

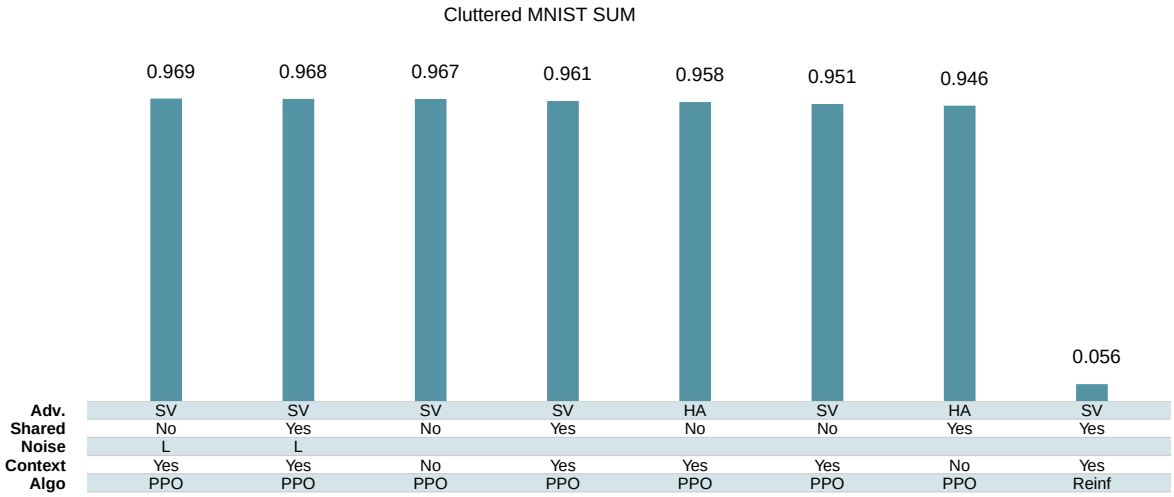
The cluttered MNIST Sum task can be decomposed in several sub-tasks, all of which are equally important for task performance. These are: finding the targets while ignoring the distractors, determine the two digits, determine the operator, get the order of the digits, combine the digits and the operator to solve the sum. The Mixed MNIST Sum task is a combination of the Cluttered MNIST Sum task and

a Cluttered MNIST task. This means that for instance the answer three can now either be the result of a sum or the classification of a single digit. The added challenge thus is to discriminate between the tasks.

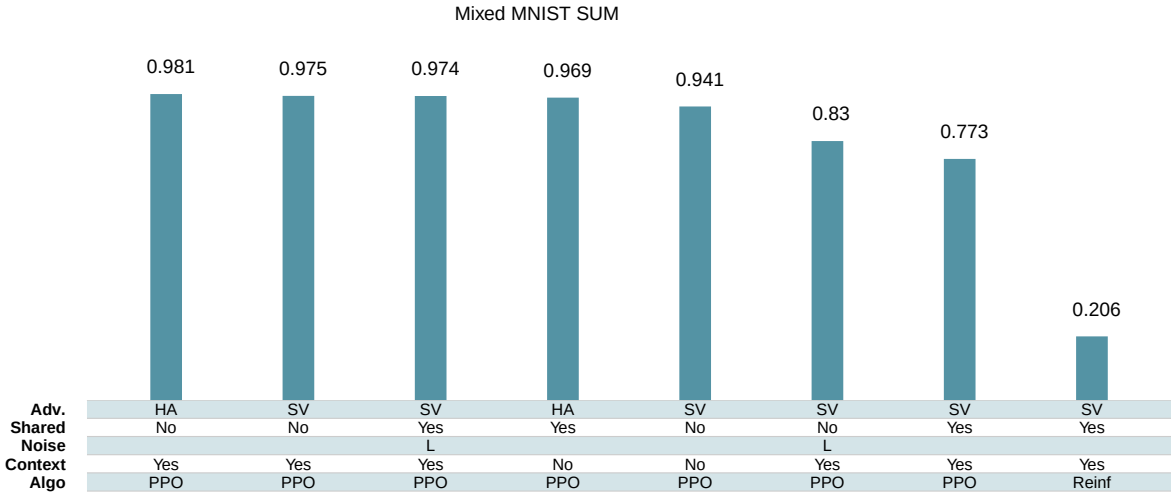
For both tasks it is not possible to directly compare the performance to similar tasks mentioned in literature. The task that comes closest is the MNIST addition tasks described by Ba et al. (2014). In their research they compared the DRAM model to a Convolutional Neural Network (CNN) on an addition task where the goal was to classify the sum of two MNIST digits. Similar to the Cluttered and Mixed MNIST Sum tasks the two MNIST characters were placed randomly on a 100x100 pixels background. There are however no distractors present and since the goal is addition only there is no operator or marking of the first digit. The scores they report are 97.5% correct for the DRAM model and 96.8% correct for the CNN.

The scores reached by Cluttered and Mixed MNIST Sum tasks can be seen in figure 5.3. What is exciting to see is that the majority of the models have reached such high scores on both tasks. The highest accuracy score on the Cluttered Sum task is 97%, while the highest score on the Mixed Sum task, with 98.1% is even higher. For the models having accuracy scores of 94% and up it is safe to assume that they have correctly learned the task or the mix of tasks. This in contrast to the model that was trained with the Reinforce algorithm. This model did not manage to converge on the Cluttered MNIST Sum task and was only able to get a low score of 20% accuracy on the Mixed MNIST Sum task. Even though degraded performance for this model was expected, given the difference between PPO and Reinforce shown for the Cluttered EMNIST task, the amount by which was not. What is even more striking, is the fact that this model was added to the selection of eight models because it resembles the DRAM model. These results do not mean that Ba et al. (2014) would not have been able to reach convergence on the task, since their paper is not enitrely clear about their exact setup, but still it is surprising to see that is exactly that model which stays so much behind.

Notable is also the fact that the highest scoring model on the Cluttered MNIST Sum task only manages to reach an accuracy score of 83% on the Mixed MNIST Sum task. An even lower accuracy score of 77.7% was reached by a model that can be described as a PPO version of the DRAM model.



(a)



(b)

Figure 5.3. Mean validation scores on the *(a)* MNIST SUM task and the *(b)* Mixed Sum task. Table under the bar plots indicates the configuration used

At this moment it is unclear why exactly these two models are showing degraded performance. Both models use the State-Value baseline, which had the advantage on the previous tasks, and both have a context layer. Especially the context layer was expected to be beneficial here, because the context information can be used to determine which task is at hand. The fact that the highest two scoring models on the Mixed SUM Task are the two models without a context layer is therefore an unexpected result.

Because the Mixed MNIST Sum Task is a random mixture of two tasks, it would be interesting to see if there is a difference in accuracy scores between the two tasks. Figure 5.4 shows the results separated

Figure 5.4. Scores on the Mixed MNIST Sum task, divided on sum or digit classification.

for both tasks. As expected all the models with overall accuracy scores above 95% show hardly any difference between performance on both tasks. The models scoring lower all show better performance on the MNIST classification task.

5.2.2 Policy Evaluation

It is safe to assume that (most) models were able to learn the required task. Because it is still not clear what strategies the models were using to perform those tasks, the remainder of this chapter is dedicated to this question. First the policies used for the Translated and Cluttered EMNIST tasks will be evaluated before the Cluttered and Mixed MNIST strategies will be discussed.

Translated & Cluttered EMNIST Tasks

It is expected that for both tasks the focus behavior over time will be different. This is because for both tasks the challenge of finding the most valuable information is a different one. In the case of the Tanslated EMNIST task, the enlarged EMNIST character and the absence of distractors means that finding it on a 100x100 pixels background should be more easy than for the Cluttered EMNIST task. Identifying a character on the other hand may be more challenging for the Translated EMNIST task. The windows sizes and number of glimpses used for both tasks are the same, meaning that a single

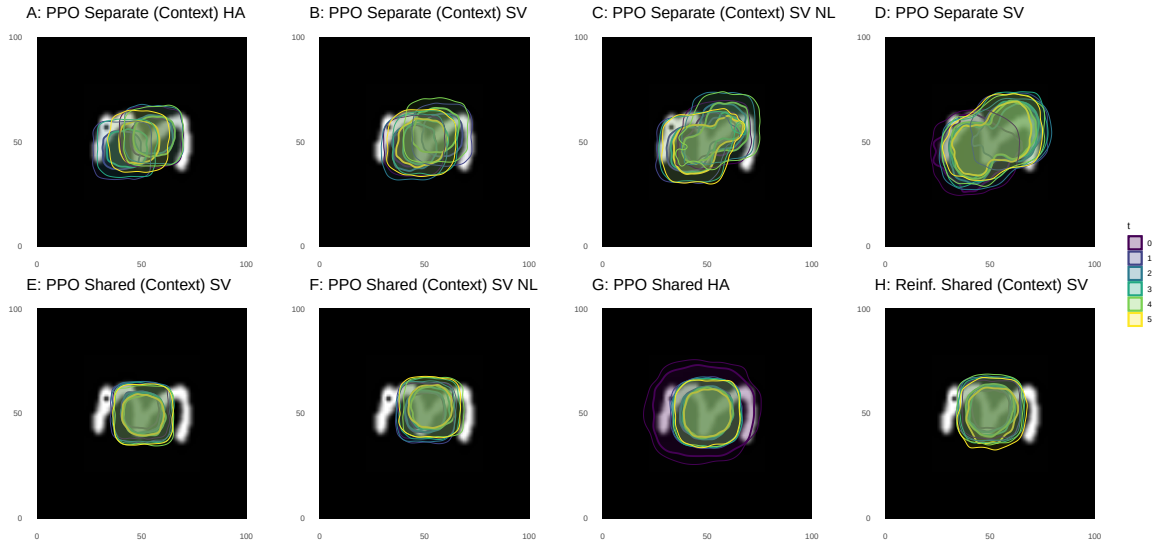


Figure 5.5. Contour plots showing focus behavior per time step for the translated EMNIST task. Contours indicate areas that contain 90% of all focus locations. Background images are present to indicate the shape of the target characters used in both tasks.

glimpse location can cover less of the target character in the Translated EMNIST task, compared to the Cluttered EMNIST task.

To gain more insight in the focus behavior over time for both experiments, 10.000 validation stimuli were used per model to collect the 6 glimpse locations together with the actual target locations. All glimpse locations are compensated for the random character placement by centering the target characters. Data is then split by the 6 time steps. This is followed by the creation of two 100x100 matrices for each time step which represent focus locations for the normal window size of 12x12 and the zoom level of 24x24. These matrices are then updated so that their rows and columns represent the number of times a coordinate falls within the glimpse window.

A kernel density estimation was applied to those matrices to get an estimation of the probability density function. This in turn was used to identify areas which have a 90% probability of being focused on. The resulting areas are then plotted for the normal window size and the zoom level. A background image with a centered EMNIST character is added to indicate the center location and the shape of the EMNIST characters used.

The contour plots for the Translated EMNIST task, see figure 5.5, first indicate that for all the eight models focus was mainly on the target character. There is however a difference in how much of the

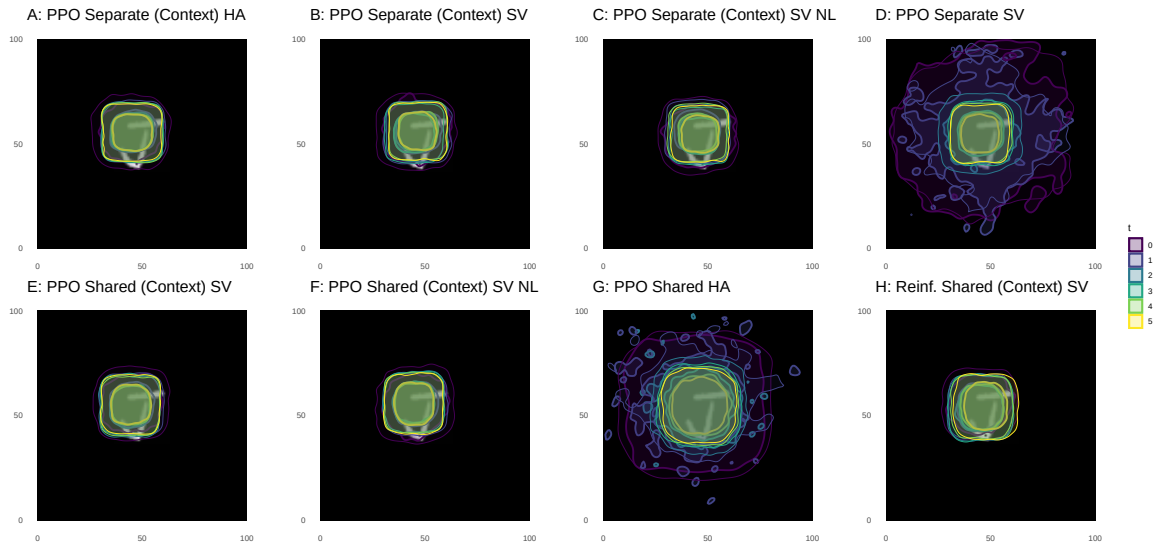


Figure 5.6. Contour plots showing focus behavior per time step for the cluttered EMNIST task. Contours indicate areas that contain 90% of all focus locations. Background images are present to indicate the shape of the target characters used in both tasks. Letters correspond to the following configurations:

target is actually covered. All the models with separate parameters seem to use a larger area than the ones with shared parameters. This however does not mean that performance improves when a larger part of the character is seen. In fact the four models having shared parameters all have higher validation scores. Even more, out of the 20 highest scoring models mentioned in table A.1 only three have shared parameters.

Model D covers the largest surface and this is also a model without a context. Without the guidance of a context, using a larger surface was expected. Model G on the other hand, which also lacks a context, does not seem to need a much larger area. Only for the first glimpse a slightly larger area can be observed.

The contour plots for the Cluttered EMNIST task, see 5.6, have two notable patterns. First for the models with a context all the focus areas are similar, indicating that they were perfectly able to focus on the target character despite the distractors. Second, the models without a context layer need the first time steps to locate the target. As mentioned the challenge of finding the target character is harder for the Cluttered EMNIST task and therefore it is not surprising to see the benefit of having context guidance here.

Table 5.2

Mean Frobenius Distance $\overline{D_F}$ and standard deviation over all possible pairs of character matrices.

					Translated EMNIST		Cluttered EMNIST	
Algorithm	Context	Noise	Shared	Advantage	$\overline{D_F}$	SD	$\overline{D_F}$	SD
PPO	Yes	No	No	Horizon-Aware	1801.19	904.94	2161.48	1037.39
PPO	Yes	No	No	State-Value	1835.87	810.99	2144.01	1014.43
PPO	Yes	Yes	No	State-Value	1427.79	560.27	2193.88	983.78
PPO	No	No	No	State-Value	1033.65	344.96	1342.36	559.68
PPO	Yes	No	Yes	State-Value	3809.53	1995.66	2579.81	1221.37
PPO	Yes	Yes	Yes	State-Value	3232.23	1572.90	2737.06	1201.54
PPO	No	No	Yes	Horizon-Aware	2294.94	1176.37	1797.06	753.36
Reinforce	Yes	No	Yes	State-Value	3758.27	1752.68	2865.81	1367.30

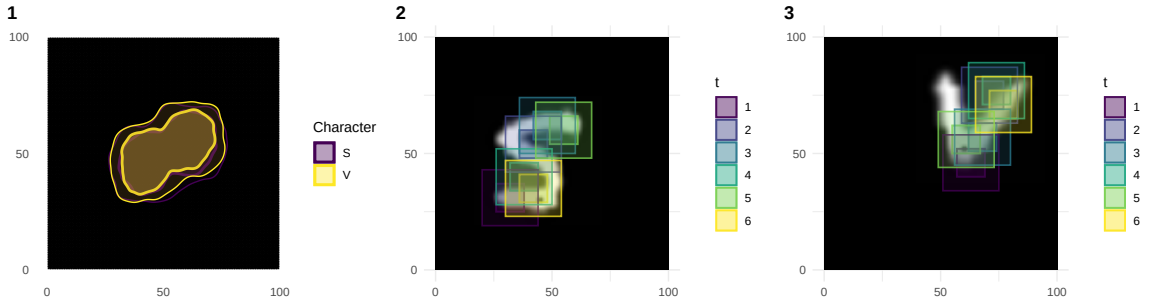
The question that arises is whether the models learn different strategies for each character or whether they are using fixed paths. For instance, for the Translated EMNIST task, It may be expected that given the larger contour areas and the absence of a context layer, model D will have more differences in focus paths than one of the shared models with a context layer.

To answer this question, balanced matrices were constructed using a similar procedure as was used for the contour plots. This time, instead of splitting on the time step, matrices were created for each character. The Frobenius norm was calculated for all the differences between all possible pairs of matrices. This value serves as a distance measure between two matrices and is calculated as follows:

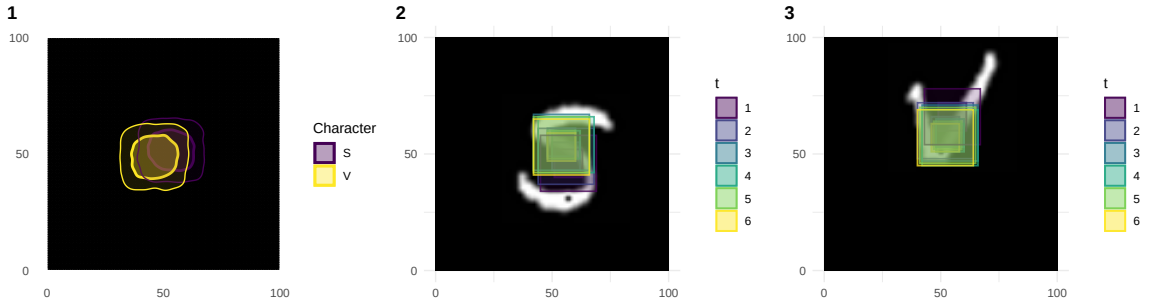
$$||A - B||_F = \sqrt{\sum_{i=1}^{m}\sum_{i=1}^{n}(A_{ij} - B_{ij})^2} \tag{5.1}$$

The overall mean and standard deviation per model are summarized for both the Translated and Cluttered EMNIST tasks in table 5.2.

For the Translated EMNIST task there is again a clear distinction between models with shared and separate parameters. The models with shared parameters all have a larger overall distance between the characters and a larger standard deviation. The models with the smallest surface areas in the contour plots have the highest values. The other way around is also true, the models with the largest surfaces have the smallest means and standard deviations. These results show that the shared parameter models are using more diverse strategies. The fact that their contour plots have smaller focus areas can



(a) Model: D PPO - Separate State-Value



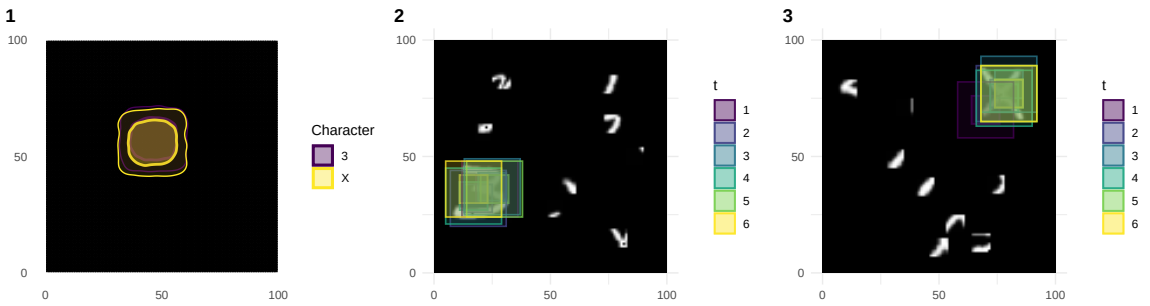
(b) Model E: PPO - Shared (Context) State-Value

Figure 5.7. Different policy behaviors on the Translated EMNIST task. *Top* Model D, which has learned a fixed glimpse pattern for all characters compared to *bottom* where the model has learned a specialized focusing pattern for each character.

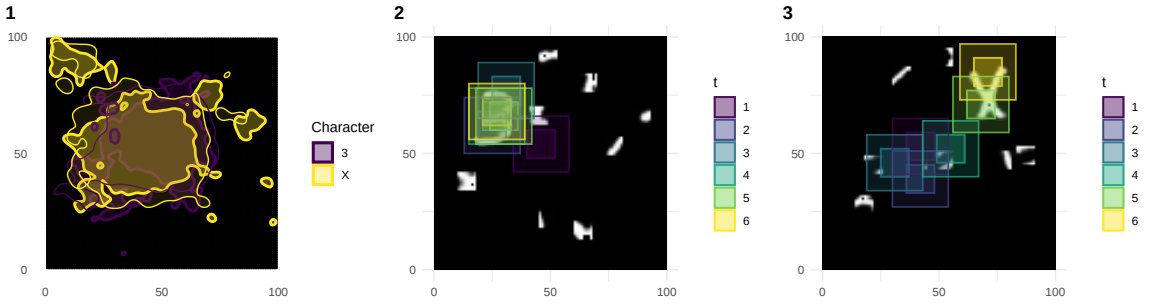
only mean that they have developed a strategy to focus only on a small, meaningful part of the target character. The models with separate action and policy networks have developed a strategy that is less tuned to the observed character, but is more a one-size-fits-all strategy.

Figure 5.7 shows an example of focus behavior for the models with the highest (E) and smallest (D) mean distance on the character S and V. These characters were selected because their calculated distance for model E is one of the largest overall. The contours for model D show the same large area as in figure 5.5 and almost perfectly overlap, while the contours for model G shows small areas for both characters with a small shift in focus. Subplots 2 and 3 give actual uncorrected examples of focus locations taken from the validation dataset. These are perfectly in line with the observation that the shared models have learned to focus on a single important region, while the separate models are using a strategy to cover most of the target character.

Table 5.2 is less clear about the difference in strategies used for the Cluttered EMNIST tasks. This is not unexpected, because of the larger part of the target that can be captured with a single glimpse. It



(a) Model B: PPO - Separate (Context) State-Value



(b) Model G: PPO - Shared Horizon-Aware

Figure 5.8. Different policy behaviors on the Cluttered EMNIST task. *Top* Model B, which shows the benefit of having a context layer to locate the target *bottom* Model G, which lacks context guidance and therefore needs to explore before it can find the target.

was already clear from figure 5.6 that the models without a context layer have more difficulty in finding the target. This may explain why these two models have the lowest overall mean distance and standard deviation. Using more glimpses for locating means that less glimpses will be available for fine-tuning.

An example of the difference between a model with and without a context layer is shown in figure 5.8. In this case models B and G are compared on the character 3 and X. Model B with a context has no problem finding the target character, while model G clearly needs several glimpses before it is able to locate the target.

Cluttered MNIST Sum

After establishing that, except for the Reinforce model, all of the trained models on both tasks are capable of showing very good performance on the Cluttered MNIST Sum task, again the question is what strategies did these models use. Heatmaps were created to investigate the overall focus behavior using data from 10000 validation samples. Creating such heatmaps is not a trivial tasks, since both

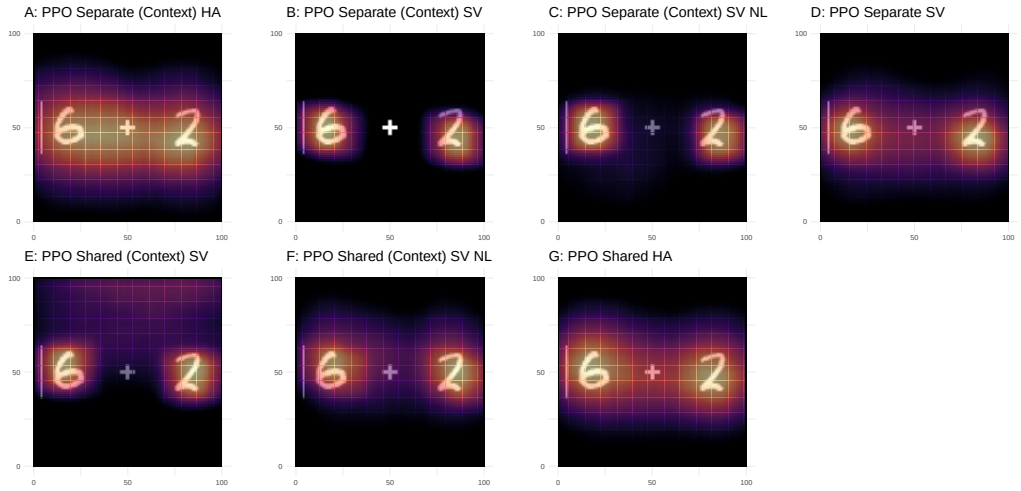
digits and the operator are randomly placed. To compensate for the random placement, it was first necessary to record the actual locations of the digits and operators used in the validation samples. However using this information to correct for the random placement can only be done when you know which of the targets the focus was on. To solve this problem, for each of the 8 glimpse locations the euclidean distance between that location and the digits and operator were calculated. The target for which the distance is smallest is then pointed out as being the focus target.

When each time step has a focus target, the glimpse locations will be shifted in such a way that the first digit in the sum is always placed at coordinate (x=18,y=50), the operator is at the center (x=50,y=50) and the right hand side digit is set at (x=82,y=50). The resulting heatmaps for the Cluttered MNIST Sum task can be seen in figure 5.9a. The model using the Reinforce algorithm that failed to converge is not shown.

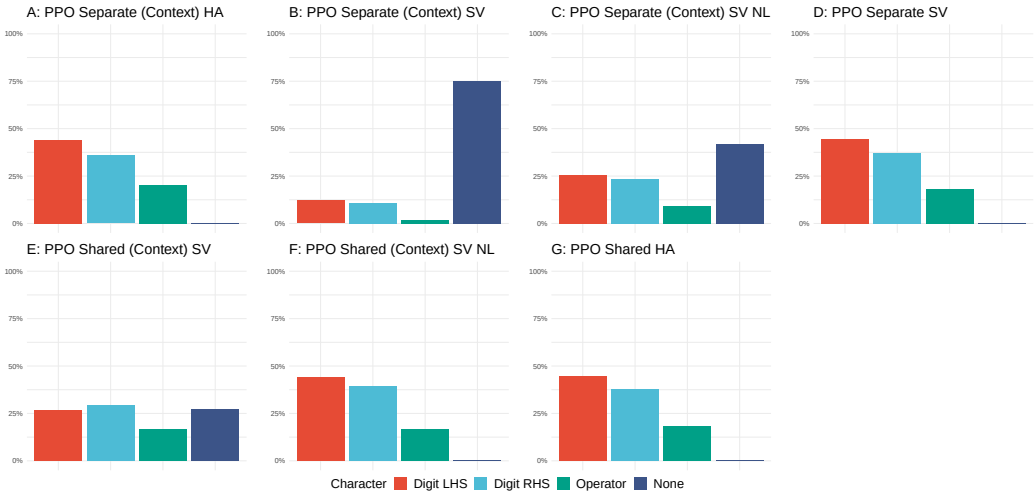
During the creation of the heatmaps a problem arose. Many of the glimpse locations are outside the image and are actually so far off that even the zoom level of 48x48 pixels is not on the image. For these situations determining the closest target is not legit and were all discarded when creating the heatmaps. This step was necessary, since compensating for a target character could have led to a shift to a location within the image.

Because the number of glimpses placed outside the image is so large for certain models, is was decided to further investigate why this happens. Figure 5.9b was created to show the percentage of glimpse locations that are either focused on the digits, operator, or outside the image. Both figures reveal that especially model B shows surprising behavior. For this model 75% of all glimpse locations are placed outside the image. By investigating the raw data it became clear that for model B, all focus locations after the second glimpse are placed outside the image and the first two glimpses are almost exclusively focusing on the digits. It seems that the operator is completely ignored. Reaching a score of 96% by only focusing on the two digits seems impossible.

Model B is not the only model that exhibits this type of behavior. Model C places most of it's glimpse locations at $t = 2, 4, 6, 7$ (starting at $t = 0$), outside the image and model G does something similar at



(a)



(b)

Figure 5.9. Top: Heatmaps highlighting the most focused locations. *Bottom:* Barplots showing the percentage of the time focus was on either the digits, operator or outside the image.

$t = 2, 3, 4, 5$ albeit to a lesser extend. It is unlikely that these models perform so well without knowing about the operator, thus something different must be going on. What all these models have in common is that they have a context layer. It may be that the models are using the context network to signal the action network about the operator. There is no information directly flowing from the context layer to the action network, but since the glimpse network also receives the focus location it may be that the focus location is used for this purpose.

If this is the case, then focus locations must be different for the plus and minus operator and should follow a distinctive pattern for each operator. To determine if this is true, a principal component analysis (PCA) was performed on the raw x and y locations recorded from the validation samples.

The reason for doing a PCA is that when there are two distinctive patterns for the operator, the x and y variables which have values corresponding to locations outside the image should exhibit strong correlation. When the PCA finds a component that can explain a large proportion of variance, it will be a strong indication that the models are using the locations outside the image for a purpose.

When performing the PCA on model B, it reveals a single component that explains 74% off all variance. When the PCA is performed on the data without the x and y variables belonging to the first two time steps the proportion of variance explained by that single factor rises to 97.6%. Further investigation shows that the distribution of the principal component values have a clear distinction in positive and negative values. By replacing the principal component with a binary variable indicating a positive or negative value the link between the operator and focus locations is immediately clear.

Table 5.3 shows the results when this procedure is applied to models B,C and E. For model B, 44.8% + 47.3% = 92.1% of the operators can be predicted this way. Similar 92.8% of the operators can be predicted for model C and even 97.5% for model E. Clearly these models use a similar strategy of focusing outside the image to signal the operator to the action network.

By applying the PCA procedure to all the models, another surprise was found. For model A, a principal component was found that explains 99% of all variance in x and y for all time steps. This factor has no direct relationship to the operator. Given that the distribution of this factor also shows a negative and a positive peak, it must mean that it has learned two distinctive glimpse patterns. That this is true can be seen in figure 5.10 which shows focus areas for each time step based on uncorrected data. By using the two patterns, the model is almost able to see the whole image.

Table 5.3

Percentage Plus or Minus Operator for a Positive or Negative PC value

PC1	Operator	Model B	Model C	Model E
Positive	Minus	44.8%	43.9%	48.4%
Positive	Plus	2.2%	0.6%	0.4%
Negative	Minus	5.7%	6.6%	2.1%
Negative	Plus	47.3%	48.9%	49.1%

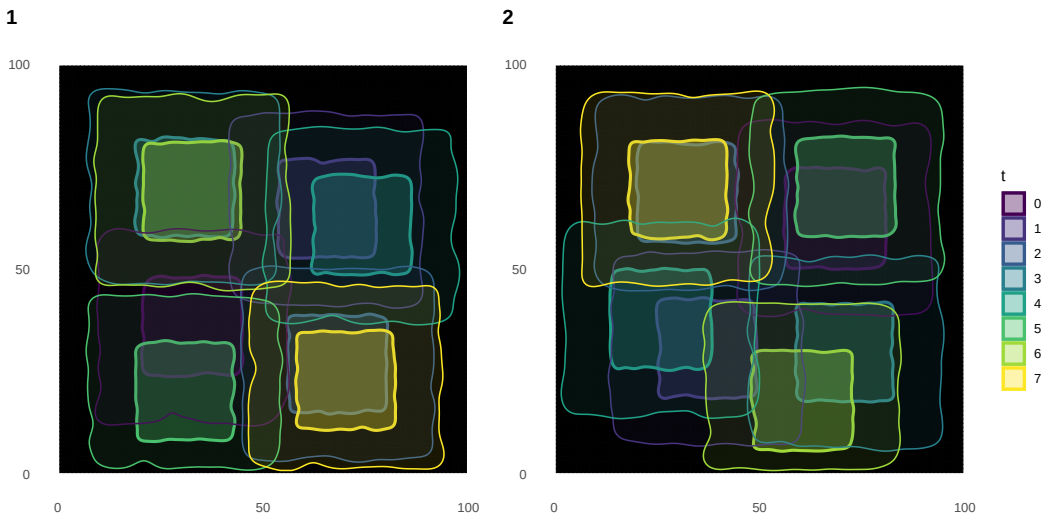


Figure 5.10. Contour plots showing the two distinct glimpse sequences as discovered after PCA for model A. *Left:* Pattern when the principal component values are negative. *Right:* Pattern for positive principal component values.

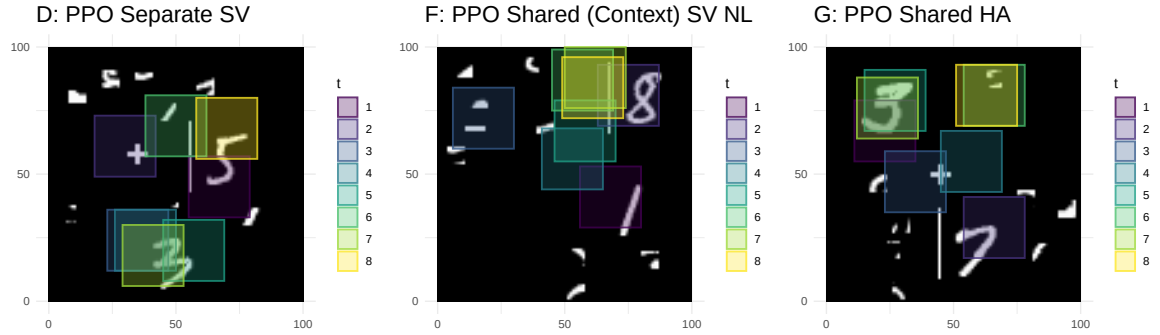


Figure 5.11. Example glimpse trajectories for the models that are using an active search strategy to solve the sums. For better visibility the zoom window of 48x48 pixels is not shown.

The remaining models D, F and G seem to behave more like you would expect from an active sensing strategy, namely actively searching for the digits and the operator. For each of the three models an example trajectory can be seen in figure 5.11

To get an idea of how strong the influence of the context layer is on the actual results, it was decided to handicap the three networks with separate parameters and a context layer. Three situations were tested. First validation accuracy scores where collected with the policy glimpse network receiving only empty image patches. Second, same empty image patches, but policy glimpse network also receives random focus locations. Third policy glimpse network in the same way handicapped on image patches and locations, but now the action network is also receiving random focus locations, but correct image

Table 5.4

Handicapped Accuracy scores, when handicapped on Policy Glimpse g(pol), Policy Location l(pol) and, Action Location l(action)

Model	g(pol)	g(pol) + l(pol)	g(pol) + l(pol) + l(action)
A	0.95	0.82	0.81
B	0.95	0.63	0.36
C	0.97	0.96	0.7

patches. When the policy network is using a combination of the context and the previous focus location, then the accuracy scores should not change when the image patches are empty. If this is true then performance should degrade when the focus locations is set to a random value. By also randomizing the focus location for the action network, the focus location cannot be used anymore to determine the operator.

Table 5.4 shows the accuracy scores with the three handicapping situations. None of the three models show degraded performance when they are not receiving the image glimpses. When the policy is receiving completely distorted information from the glimpse network, classification performance goes down for models A and B, but not for C. When the action network cannot use the glimpse locations for classification a drop in performance is observed for model B and C.

It is clear that for these three models the visual information processed by the context is sufficient for planning focus locations. Surprisingly, model C is able to plan subsequent actions without glimpse information. The drop in performance for models B and C and not for A when the possibility to signal the operator was removed, is more evidence that models B and C are indeed using such a strategy.

Another interesting question is whether the models have learned a fixed sequence of digits and operator to solve the sums. For instance number left-hand side (LHS), operator, number right-hand side (RHS) as is common for humans.

Figure 5.12 shows the order in which the numbers and operator are first visited. There is a clear pattern visible for all models, they all have a preference to focus at the LHS number before focusing on the RHS number. The reason that the pattern with only the LHS and RHS and not the operator has the highest scores for models B,C and E can easily be explained because of the operator signaling reported

Figure 5.12. Digit and operator focus patterns on the Cluttered MNIST Sum task. The patterns indicate the order in which a digit or operator was first visited.

above. For models D, F, and G, a possible explanation can be that the larger glimpse window of 24x24 pixels and a zoom of 48x48 pixels is large enough to sometimes capture the operator together with one of the numbers.

It must also be noted that there always is an initial glimpse from which the information is also processed. This initial glimpse is needed because the policy network can only output the next focus location based on some input. In the case of the shared parameter models, this initial glimpse location passes both the recurrent networks for the action and policy network. Those models can thus combine the initial glimpse with the policy glimpses for classification. In some occasions these initial glimpses may contain information about the presented targets.

The models with a separate action and policy network do not have this advantage, the initial glimpse is only used by the policy network.

The only real surprise is model A, for which was shown that it only uses two fixed glimpse patterns. So far the principal component found for model A was unexplained, but the most plausible explanation is that the glimpse pattern is chosen in such a way that the order of the numbers is LHS, RHS. Again a type of behavior that can only be explained when actions are guided by a context.

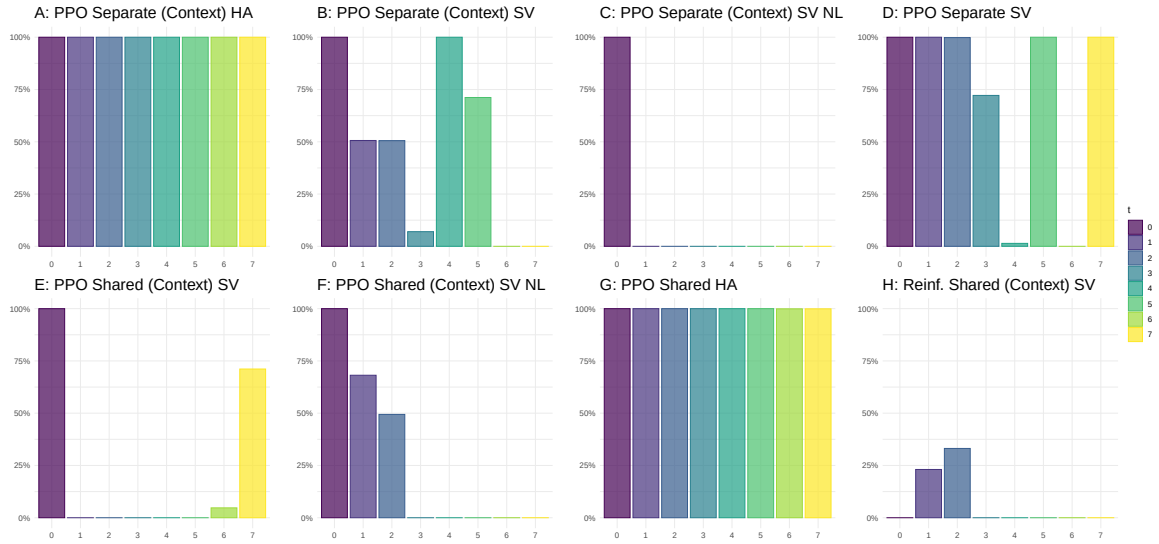


Figure 5.13. Percentage of focus locations on the image per time-step.

Mixed MNIST Sum Task

Because the Mixed MNIST Sum task consists of two tasks, the first question to ask is what influence does this have on the developed strategies. Is it a combination of the strategy used for the Cluttered MNIST Sum task and the Cluttered EMNIST task or did the models develop a new strategy? As can be seen in figures 5.3 and 5.4 five models were performing good, two showed degraded performance, and the Reinforce model only managed to get an above chance score on the MNIST classification part of the task.

Again many of the focus locations are placed outside the image. Figure 5.13 shows the distribution of focus locations on the image per time-step. Only models A and G are focusing purely on the image. The bad performance of the Reinforce model is caused because only a small proportion of glimpses are actually focused on the image. More interesting are models C and E. Both were mentioned as models that were using an operator signaling mechanism on the Cluttered MNIST Sum task. Model C seems to be using just a single glimpse, while model E is using at most 2.

A PCA was again performed on both models, but this time a clear identifiable component could not be found. For model E it might have been the case that it was using two glimpses when a sum was presented and one for a single digit, but this is not true. In 42% of the cases focus is on the image for

the last glimpse when presented a sum and 99.9% for a single number. The only possible explanation is that the two models learned a policy that only focuses on a single digit. For the MNIST classification task this is enough for good performance, but for the SUM task it is not.

The reason the two models still have reasonable performance can be explained by two facts. First the glimpse window of 24x28 pixels and a zoom of 48x48 pixels may be enough to collect enough of the two digits when they are placed close enough to each other. Since it was shown for the Cluttered MNIST Sum task that these models were able to use operator signaling it was probably used here also, even though analysis was not able to detect it.

Model B on the other hand is still clearly using the signaling strategy. When the x and y locations of the last two glimpses are replaced by a single principal component, using the same positive and negative indicator variable, it was able to predict the operator in 99.9% of all cases where a sum was presented. Apart from that on average this model is using two glimpses for the MNIST classification task and three for the MNIST Sum task. Thus it seems that this model has managed to keep the operator signaling strategy to process the sums, but has managed to combine it with a classification strategy.

Model A, which was using just two fixed focus patterns does not seem to be using the same strategy for the Mixed MNIST Sum Task. PCA analysis does not show a single strong component, not even when split on the task. Model G is the other model with all focus locations inside the image. The raw data is showing some patterns in the data, for instance histograms of the x and y variables all show a pattern with positive and negative peaks. Strong correlations between the variables are also present, indicating that there probably are several separate patterns that were used. A distinction made on sum or classification does not show a difference.

Model F is the biggest surprise here, because it was found that this model is now also using the operator signaling mechanism. A combination of the second, third and last focus location could be replaced by a single principal component that was able to predict 98.9% of the operators used when a sum was presented. Even more surprising is the fact that focus locations for the second glimpse were all placed on the image when a single MNIST digit was presented and are all outside the image when

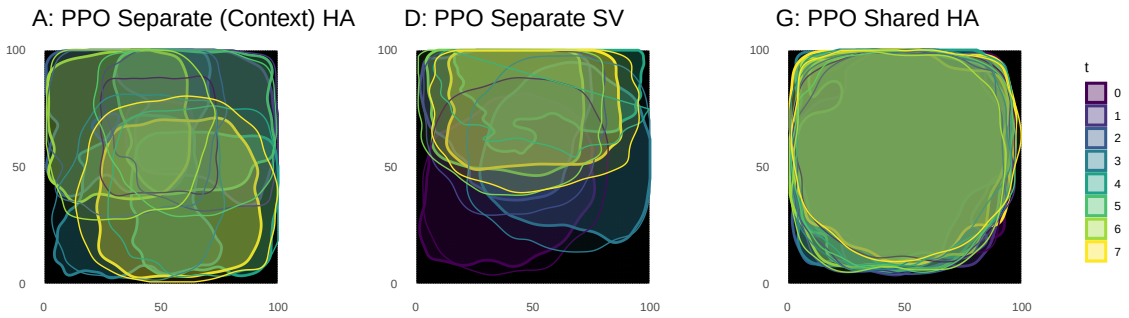


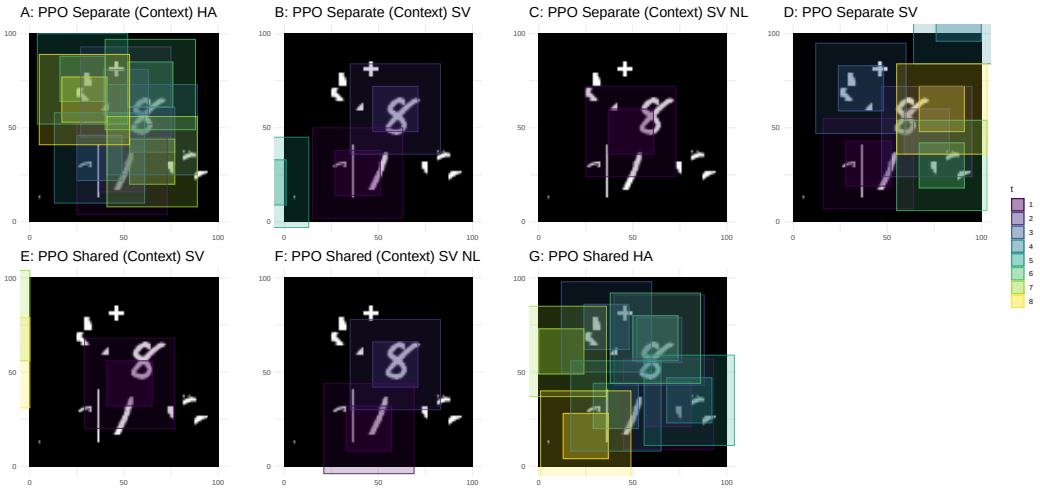
Figure 5.14. Contour plot for models A, D and F for the MNIST classification part of the Mixed MNIST Sum Task

presented a sum. This model is the most efficient in extracting valuable information, because glimpses focused outside the image only perceive meaningless noise.

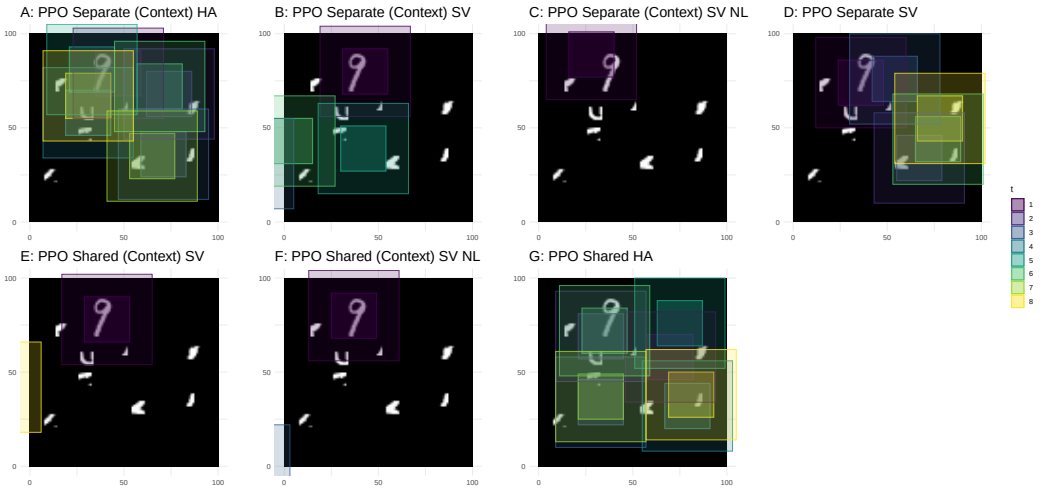
For the models (A,D,G) that focus all or most of their glimpses on the image, it is interesting to see what their focus behavior is when MNIST digits are presented. Espically the two models without context guidance should have more trouble finding out which task they need to perform. Figure 5.14 shows the contour plots per time-step for each of those models. Model A is focusing it's first glimpse on the target, but after that it is using a structured pattern where each follow-up glimpse is exploring a different region.

That models D and G show more exploration was expected. For model D it looks like only the first four time-step are actively used for exploration. Model G on the other hand seems to need all the time-steps for exploration. No clear pattern can be detected here, it behaves more like a random search.

Figure 5.15 shows example trajectories taken from the validation dataset for all the models except the one trained with the Reinforce algorithm. Both a sum and a single digit as shown. It must be noted that all seven models gave the correct answers. The examples shown support the conclusions drawn above. Models C and E are able to answer the sum by focusing between the two numbers and are most probable using the signaling method to inform the classification network about the operator. Models A is focusing on the numbers and operator, but also on some clutter when presented a sum. When a single digit is shown, the first glimpse is focused on that digit and after that it is exploring other regions of the image. The pattern shown by model D resembles that of model A. Model G is using a strategy



(a)



(b)

Figure 5.15. Example Mixed MNIST Sum Task trajectories *Top:* Model behavior compared using the same sum. *Bottom:* Model behavior compared on the same MNIST number.

that is almost covering the whole image. Finally models B and F show that, in the case of a sum they indeed focus on the two digits before they start focusing outside the image. In the case of a single digit, both models use a single glimpse to focus on that digit.

6 Discussion

Main research question was whether an active sensing strategy could be learned through reinforcement learning. As was shown in chapter 3 active sensing can be framed as a Partially Observable Markov Decision Process (POMDP), or to be more precise a belief MDP. Finding an actual solution for a belief-MDP is known to be np-hard, meaning that the actual solution can only be approximated (Hsu, Lee, & Rong, 2007).

The method used for approximating was using artificial neural networks as function approximators and train them with a Reinforcement Learning algorithm. Recurrent layers were used for integrating the glimpse states extracted from different focus locations to form the internal belief states.

The belief-MDP framework is not only useful for computational modeling of active sensing, but it is in fact a biologically plausible method for learning such a strategy. First evidence is pointing into the direction that the human brain has multiple reinforcement learning mechanisms for deciding on actions that will maximize future reward. (Lee et al., 2012; Niv, 2009; O'Doherty, Lee, & McNamee, 2015)

Second recurrent neural networks are known for being able to approximate dynamical systems, like the constantly changing belief states. Given that the brain also has a large number of recurrent connections, it is entirely plausible that the brain has similar mechanisms to approximate dynamical systems. (Kriegeskorte & Douglas, 2018; Marblestone et al., 2016)

Approximating a belief-MDP is in itself a challenge, but the difficulty was even larger for the neural networks presented in this thesis. In a standard Reinforcement Learning setup the environment, for

instance a computer game, gives a direct reward signal. By learning to maximize rewards collected from the environment, performance will improve. Things are different for the active sensing models. The action networks that are doing the classification are trained supervised together with the policy. The accuracy of the classification determines the quality of the reward signal used for optimizing the policy. In turn, the action network can only be optimized when the policy network select valuable focus locations.

Given the extra difficulties this reciprocal relationship between policy behavior and task performance adds to challenge of approximating an active sensing strategy, it was decided to use two existing models, RAM and DRAM, that have already proven themselves as base (Ba et al., 2014; Mnih et al., 2014). The biggest disadvantage of their research is that the main focus was on accuracy scores and no in-depth analysis of the policy behavior was given.

A thorough policy analysis was done in this research, but only after several possible modifications of the RAM and DRAM models were tested against the Translated and Cluttered EMNIST tasks. Motivation for this was that there seemed to be room for improving the models. The modifications tested were using PPO as Reinforcement Learning algorithm instead of Reinforce, using a different type of advantage function, enhance exploration by adding parameter space noise, and using separate policy and actions networks instead of having shared parameters. The main difference between RAM and DRAM is that the latter features a context layer. The effect of having such a context layer was also added to the test. Combined, a total of 48 possible model configurations were tested.

The results presented in section 5.1 have shown that there is one improvement option that clearly stands out above the rest. The PPO algorithm is outperforming the Reinforce algorithm when the tasks are becoming more complex. With PPO not only task performance improves, but also the number of epochs needed for converging is much lower. This can be explained by looking at the main improvements PPO brings over Reinforce. PPO is improving sample efficiency by importance sampling from the current policy, while the clipping parameter prevents it from deviating too much from the current policy. (Schulman et al., 2017). This ensures that compared to Reinforce more information from

a given sample will be used for policy improvement. At the same time, too large policy updates that may steer the policy in a wrong direction are prevented. It can be concluded that this is advantageous in the active sensing setup where the reward signal, especially in earlier training phases. can be quite unreliable.

Later on during the evaluation of the policies learned for the Cluttered MNIST Sum and Mixed MNIST Sum tasks, see section 5.2.1, the single model trained using Reinforce was not even able to reach a decent score on the tasks.

A second modification whose effect was tested, was that of the Horizon-Aware baseline. Even though variance in the reward signal was already reduced by using the 0,1 indicator function, the unreliable nature of the reward signal will cause unwanted variance. A good baseline should be able to reduce this variance further and therefore it was decided to explore the Horizon-Aware advantage function developed by Tucker et al. (2018). According to their work, it should outperform the State-Value advantage used by the RAM and DRAM models.

The results however did not show such an advantage when used on the Translated and Cluttered EMNIST tasks. Later when evaluating the MNIST Sum and Mixed Sum tasks, the models using the Horizon-Aware baseline did stand out against the rest as models that learned a policy covering the whole image instead of actively focusing on target locations. For both tasks this did prove to be a successful strategy as the high accuracy scores proved. Good performance however does not mean that the strategies used by these two models can be regarded as active sensing. It can be argued that the models have learned that all the task relevant information will be somewhere within the stimulus image, but it cannot be said that they were actively focusing on the most meaningful parts.

Adding a context layer also had a profound effect. Not on the translated EMNIST task, where the need for guidance by a system that has an overview was smallest. But certainly present in models trained for the other three tasks. In the case of the Cluttered EMNIST task, where finding the target amongst distractors was the biggest challenge, models without a context layer clearly had more difficulty

finding the target. This was also translated in lower accuracy scores as the estimation plots in figure 5.1 have shown.

For the Cluttered MNIST Sum and Mixed MNIST Sum tasks the effects of having a context was even more profound. Most notable are the models that use a mechanism where focusing on a certain location outside the stimulus image gives a message to the action network about the type of operator that was used. For the models with a separate action and policy network, it was even shown on the Cluttered SUM task that they could perform equally well with a policy that is not even receiving the image patches. Performance of these models did degrade when operator signaling was made impossible by handicapping the action networks.

Some may regard adding a context layer as cheating, but it can also be argued that it is a biologically plausible mechanism. Research has shown that the human brain uses peripheral vision, which has a much larger field of view than foveal vision, to extract the gist of a scene. A gist can be extracted quickly from contextual information and is known to have an immediate effect on the planning of eye movements when foveal information is not available (Castelhano & Henderson, 2007; Hillstrom, Scholey, Liversedge, & Benson, 2012; Oliva & Torralba, 2007)

Adding parameter space noise to enhance exploration did not seem to bring much improvement. The parameter space noise option developed by Plappert et al. (2017) even led to a number of unstable models that were not able to converge. Results for the noise-layer developed by Fortunato et al. (2017) were less dramatic, but no added benefit could be observed. Despite the fact that the exploration options did not deliver what was hoped for, it is still believed that active sensing models will benefit from enhanced exploration. A promising direction may be to enhance the models with curiosity. This adds an internal drive to sample information about which uncertainty is still high. Such an active mechanism of reducing uncertainty about the partially observed state will almost certain lead to advanced exploration strategies that will help in developing active sensing strategies for even more complex tasks than described in this thesis. Adding curiosity is not without difficulties, for instance

what to do with random noise that will always lead to high uncertainty. (Fu, Co-Reyes, & Levine, n.d.; Gottlieb & Oudeyer, 2018; Pathak, Agrawal, Efros, & Darrell, 2017).

The last modification that was tested was separating the action and policy networks. The results are not immediately clear about whether this is beneficial or not. On the Translated EMNIST task having separate networks was a disadvantage. Scores for models with shared parameters were higher and the policies used by the shared parameter models were better in finding parts of the target character that were most valuable for classification. On the Cluttered EMNIST task, no differences in either accuracy scores or policy behavior could be observed. The policies observed when analyzing the Cluttered and Mixed Sum tasks also did not reveal a clear distinction between the two options.

The rationale behind separating the networks is the fact that for a long time now it is assumed that visual information is processed in the brain by two distinct pathways, the ventral ”what” versus the dorsal ”where”. (Ungerleider & Haxby, 1994) More recently this so-called two-visual system TVS model is heavily debated (de Haan & Cowey, 2011; de Haan, Jackson, & Schenk, 2018; Rauschecker, 2018). It was even shown that humans in fact may have three such pathways (Haak & Beckmann, 2018). Interesting is the view by Scholte et al. (2018) which state that such pathways are the result of a brain that is optimizing cost functions, similar to the way deep learning models are optimizing their cost functions.

The active sensing models in this research are also optimizing two cost functions. First by optimizing the policy for maximizing expected future reward and second by increasing classification accuracy. This can also be regarded as a distinction between ”where” to look and ”what” am I looking at. When this indeed leads to separate pathways, it can be expected that shared and separate parameter models will be showing similar behavior. It may also be the case that optimizing two cost functions in one network will lead to interference, in which case performance may be hampered and models with shared action and policy networks have an advantage.

The fact that no real distinction between the two types of models was found is supportive of the emergence of two pathways in the shared models. On the Cluttered and Mixed MNIST Sum tasks, the

shared parameter models may very well have developed mechanisms similar to the separated models, where the context and the focus locations are sufficient for a good behaving policy. Visual information extracted with the glimpses is then solely used for classification. Translated to human brains this would resemble foveal information going into the ventral stream for object recognition, while peripheral information is sent to the dorsal stream for the planning of eye movements.

This is still speculative and it would be interesting to investigate whether active sensing models with shared parameters indeed form separate pathways.

The question whether all these models have actually learned an active sensing strategy has not been answered so far. Again, see section 5.2, using the key aspects formulated by Gottlieb (2018), an active sensing strategy needs prior knowledge of the task structure, it knows how much information gain is expected when focusing at a certain location and the fact that the decision to sample at a certain location is made before discriminating.

By looking at the overall performance on the four tasks, all good performing models have learned the task structure. Whether it was finding a target character and ignoring distractors or combining digits and an operator to solve sums. Because of the diversity in the policies that emerged during task training, the question whether or not the other two key aspect were present is more difficult to answer. For the Translated and Cluttered EMNIST tasks it can most easily be argued that those key aspects were present. Most valuable information was always available at the target location and by direction attention to those locations the action networks were able to classify the EMNIST characters.

The policies learned for the Cluttered and Mixed MNIST Sum tasks have shown behavior where the operator is signaled to the action network without actually focusing on it. In those situation the policy was certainly aware that the operator was valuable for the task. However for focusing outside the image to signal the operator it is questionable whether this can be seen as knowing about the information gain associated with sampling at that location. Focusing at such a location will only sample noise, but the location in space itself does inform the action network. In that sense there is information gain associated with it.

Such signaling strategies can also be regarded as being smart strategies, since they make sure that the action networks only need to process either the digits or meaningless random noise. It prevents the processing of distractors by the action network. The distractors are small patches extracted from the MNIST digits and could have had a negative influence on classification, ignoring them altogether may in fact be a wise thing to do.

Even for models that use fixed focus sequences it can be argued that they know where the most valuable information is present. Focusing is in no way restricted to areas within the stimulus image and therefore learning a search path that can cover most of the stimulus image can be regarded as knowing where the task related information can be found. A special mention for the strategy learned by model A on the Cluttered MNIST Sum task is in place. The fact that the two search paths used to solve the sum seems to be related to seeing the left-hand side digit before the right-hand side one can also be regarded as a smart policy.

In general it was interesting to see that the models were all focusing on the marked left-hand side digit before focusing on the right-hand side digit. This is supportive of the aspect that prior knowledge of the task structure needs to be present.

Concluding, it was surprising to observe the diversity in policies used for the tasks. It can be concluded that all the good performing models found a way to extract or signal the information needed for the task at hand. In that sense they were all deploying an active sensing strategy. When comparing to human behavior, it is clear that not all strategies resemble the behavior expected from humans. It is not expected that humans are using signaling strategies for scene understanding.

A big difference here is that all the neural network models were trained on a single task or a mixture of two tasks using simple stimuli. The tasks were developed to capture important aspects of visual attention, but are nowhere near as rich as the natural environments humans operate in. Humans when performing a task may simultaneously be occupied with other tasks or have internal processes influencing the value of information. It is therefore expected that active sensing models, like the ones

presented in this thesis, will start to behave more like humans when they need to operate in natural environments and are not restricted on performing a single task.

The research presented here has already shown that active sensing policies approximated using Reinforcement Learning can lead to smart policies. The fact that the diversity in policies increased with task complexity leads to believe that scaling such models up to complex natural scenes will only lead to increasingly more intelligent policies. By adding more advanced exploration techniques like curiosity this effect can only be strengthened.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. Retrieved from https://www.tensorflow.org/

Aru, J., & Vicente, R. (2018, March 28). What deep learning can tell us about higher cognitive functions like mindreading? arXiv: 1803.10470 [cs, q-bio]. Retrieved November 26, 2018, from http://arxiv.org/abs/1803.10470

Ba, J., Mnih, V., & Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*.

Balcarras, M., Ardid, S., Kaping, D., Everling, S., & Womelsdorf, T. (2016, February). Attentional Selection Can Be Predicted by Reinforcement Learning of Task-relevant Stimulus Features Weighted by Value-independent Stickiness. *Journal of Cognitive Neuroscience*, *28*(2), 333–349. doi:10.1162/jocn_a_00894

Castelhano, M. S., & Henderson, J. M. (2007). Initial scene representations facilitate eye movement guidance in visual search. *Journal of Experimental Psychology: Human Perception and Performance*, *33*(4), 753–763. doi:10.1037/0096-1523.33.4.753

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014, June 3). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv: 1406.1078 [cs, stat]. Retrieved August 12, 2019, from http://arxiv.org/abs/1406.1078

Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017, February 17). EMNIST: an extension of MNIST to handwritten letters. arXiv: 1702.05373 [cs]. Retrieved October 23, 2018, from http://arxiv.org/abs/1702.05373

Cox, D. D., & Dean, T. (2014, September 22). Neural Networks and Neuroscience-Inspired Computer Vision. *Current Biology*, *24*(18), R921–R929. doi:10.1016/j.cub.2014.08.026

de Haan, E. H. F., & Cowey, A. (2011, October 1). On the usefulness of 'what' and 'where' pathways in vision. *Trends in Cognitive Sciences*, *15*(10), 460–466. doi:10.1016/j.tics.2011.08.005

de Haan, E. H. F., Jackson, S. R., & Schenk, T. (2018, January 1). Where are we now with 'What' and 'How'? *Cortex*. Where to Go Now with "What & How", *98*, 1–7. doi:10.1016/j.cortex.2017.12.001

Dufourq, E., & Bassett, B. A. (2017, September 26). EDEN: Evolutionary Deep Networks for Efficient Machine Learning. arXiv: 1709.09161 [cs, stat]. Retrieved July 23, 2019, from http://arxiv.org/abs/1709.09161

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., … Legg, S. (2017, June 30). Noisy Networks for Exploration. arXiv: 1706.10295 [cs, stat]. Retrieved August 1, 2018, from http://arxiv.org/abs/1706.10295

Friston, K., Adams, R., Perrinet, L., & Breakspear, M. (2012). Perceptions as hypotheses: Saccades as experiments. *Frontiers in psychology*, *3*, 151.

Fu, J., Co-Reyes, J., & Levine, S. (n.d.). EX2: Exploration with Exemplar Models for Deep Reinforcement Learning, 11.

Geisler, W. S. (2011). Contributions of ideal observer theory to vision research. *Vision research*, *51*(7), 771–781.

Gottlieb, J. (2012, October). Attention, Learning, and the Value of Information. *Neuron*, *76*(2), 281–295. doi:10.1016/j.neuron.2012.09.034

Gottlieb, J. (2018, May 1). Understanding active sampling strategies: Empirical approaches and implications for attention and decision research. *Cortex*. The Unconscious Guidance of Attention, *102*, 150–160. doi:10.1016/j.cortex.2017.08.019

Gottlieb, J., Hayhoe, M., Hikosaka, O., & Rangel, A. (2014, November 12). Attention, Reward, and Information Seeking. *Journal of Neuroscience*, *34*(46), 15497–15504. doi:10.1523/JNEUROSCI.3270-14.2014. pmid: 25392517

Gottlieb, J., & Oudeyer, P.-Y. (2018, December). Towards a neuroscience of active sampling and curiosity. *Nature Reviews Neuroscience*, *19*(12), 758–770. doi:10.1038/s41583-018-0078-0

Gregory, R. L. (1980). Perceptions as hypotheses. *Phil. Trans. R. Soc. Lond. B*, *290*(1038), 181–197.

Groen, I. I., Silson, E. H., & Baker, C. I. (2017). Contributions of low-and high-level properties to neural processing of visual scenes in the human brain. *Phil. Trans. R. Soc. B*, *372*(1714), 20160102.

Güçlü, U., & van Gerven, M. A. J. (2015, July 8). Deep Neural Networks Reveal a Gradient in the Complexity of Neural Representations across the Ventral Stream. *Journal of Neuroscience*, *35*(27), 10005–10014. doi:10.1523/JNEUROSCI.5023-14.2015. pmid: 26157000

Haak, K. V., & Beckmann, C. F. (2018, January 1). Objective analysis of the topological organization of the human cortical visual connectome suggests three visual pathways. *Cortex*. Where to Go Now with "What & How", *98*, 73–83. doi:10.1016/j.cortex.2017.03.020

Hayhoe, M., & Ballard, D. (2014, July 7). Modeling Task Control of Eye Movements. *Current Biology*, *24*(13), R622–R628. doi:10.1016/j.cub.2014.05.020

Henderson, J. M., & Hayes, T. R. (2017). Meaning-based guidance of attention in scenes as revealed by meaning maps. *Nature Human Behaviour*, *1*(10), 743.

Henderson, J. M., & Hayes, T. R. (2018). Meaning guides attention in real-world scene images: Evidence from eye movements and meaning maps. *Journal of Vision*, *18*(6), 10–10.

Hillstrom, A. P., Scholey, H., Liversedge, S. P., & Benson, V. (2012). The effect of the first glimpse at a scene on eye movements during search. *Psychonomic Bulletin & Review*, *19*(2), 204–210.

Ho, J., Tumkaya, T., Aryal, S., Choi, H., & Claridge-Chang, A. (2019, July). Moving beyond P values: data analysis with estimation graphics. *Nature Methods*, *16*(7), 565. doi:10.1038/s41592-019-0470-3

Hochreiter, S., & Schmidhuber, J. (1997, December 1). Long Short-term Memory. *Neural computation*, *9*, 1735–80. doi:10.1162/neco.1997.9.8.1735

Hsu, D., Lee, W. S., & Rong, N. (2007). What Makes Some POMDP Problems Easy to Approximate? In *Proceedings of the 20th International Conference on Neural Information Processing Systems* (pp. 689–696). NIPS'07. Vancouver, British Columbia, Canada. USA: Curran Associates Inc. Retrieved August 12, 2019, from http://dl.acm.org/citation.cfm?id=2981562.2981649

Jayasundara, V., Jayasekara, S., Jayasekara, H., Rajasegaran, J., Seneviratne, S., & Rodrigo, R. (2019, January). TextCaps: Handwritten Character Recognition With Very Small Datasets. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 254–262). 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). doi:10.1109/WACV.2019.00033

Kingma, D. P., & Ba, J. (2014, December 22). Adam: A Method for Stochastic Optimization. arXiv: 1412.6980 [cs]. Retrieved August 12, 2019, from http://arxiv.org/abs/1412.6980

Koehler, K., & Eckstein, M. P. (2017). Beyond scene gist: Objects guide search more than scene background. *Journal of Experimental Psychology: Human Perception and Performance*, *43*(6), 1177.

Kowler, E. (2011). Eye movements: The past 25 years. *Vision research*, *51*(13), 1457–1483.

Kriegeskorte, N. (2015). Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annual Review of Vision Science*, *1*(1), 417–446. doi:10.1146/annurev-vision-082114-035447

Kriegeskorte, N., & Douglas, P. K. (2018, September). Cognitive computational neuroscience. *Nature Neuroscience*, *21*(9), 1148–1160. doi:10.1038/s41593-018-0210-5

Kuzovkin, I., Vicente, R., Petton, M., Lachaux, J.-P., Baciu, M., Kahane, P., … Aru, J. (2018, August 8). Activations of deep convolutional neural networks are aligned with gamma band activity of human visual cortex. *Communications Biology*, *1*(1), 107. doi:10.1038/s42003-018-0110-y

Lee, D., Seo, H., & Jung, M. W. (2012). Neural Basis of Reinforcement Learning and Decision Making. *Annual Review of Neuroscience*, *35*(1), 287–308. doi:10.1146/annurev-neuro-062111-150512. pmid: 22462543

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., … Wierstra, D. (2015, September 9). Continuous control with deep reinforcement learning. arXiv: 1509.02971 [cs, stat]. Retrieved July 17, 2018, from http://arxiv.org/abs/1509.02971

Marblestone, A. H., Wayne, G., & Kording, K. P. (2016). Toward an Integration of Deep Learning and Neuroscience. *Frontiers in Computational Neuroscience*, *10*. doi:10.3389/fncom.2016.00094

Mnih, V., Heess, N., & Graves, A. (2014). Recurrent models of visual attention. In *Advances in neural information processing systems* (pp. 2204–2212).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015, February). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. doi:10.1038/nature14236

Najemnik, J., & Geisler, W. S. (2005). Optimal eye movement strategies in visual search. *Nature*, *434*(7031), 387.

Najemnik, J., & Geisler, W. S. (2009). Simple summation rule for optimal fixation selection in visual search. *Vision research*, *49*(10), 1286–1294.

Niv, Y. (2009, June 1). Reinforcement learning in the brain. *Journal of Mathematical Psychology*. Special Issue: Dynamic Decision Making, *53*(3), 139–154. doi:10.1016/j.jmp.2008.12.005

O'Doherty, J. P., Lee, S. W., & McNamee, D. (2015, February 1). The structure of reinforcement-learning mechanisms in the human brain. *Current Opinion in Behavioral Sciences*. Cognitive Control, *1*, 94–100. doi:10.1016/j.cobeha.2014.10.004

Oliva, A., & Torralba, A. (2007). The role of context in object recognition. *Trends in cognitive sciences*, *11*(12), 520–527.

OpenAI. (2018). OpenAI Five. Retrieved from https://blog.openai.com/openai-five/

Pascanu, R., Mikolov, T., & Bengio, Y. (2012, November 21). On the difficulty of training Recurrent Neural Networks. arXiv: 1211.5063 [cs]. Retrieved August 12, 2019, from http://arxiv.org/abs/1211.5063

Pascanu, R., Mikolov, T., & Bengio, Y. (2013, February 13). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning* (pp. 1310–1318). International

Conference on Machine Learning. Retrieved June 22, 2019, from http://proceedings.mlr.press/v28/pascanu13.html

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017, July). Curiosity-Driven Exploration by Self-Supervised Prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 488–489). 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). doi:10.1109/CVPRW.2017.70

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., … Andrychowicz, M. (2017, June 6). Parameter Space Noise for Exploration. arXiv: 1706.01905 [cs, stat]. Retrieved May 26, 2019, from http://arxiv.org/abs/1706.01905

Rauschecker, J. P. (2018, January 1). Where, When, and How: Are they all sensorimotor? Towards a unified view of the dorsal pathway in vision and audition. *Cortex*. Where to Go Now with "What & How", *98*, 262–268. doi:10.1016/j.cortex.2017.10.020

Rayner, K. (2009, August 1). Eye movements and attention in reading, scene perception, and visual search. *The Quarterly Journal of Experimental Psychology*, *62*(8), 1457–1506. doi:10.1080/17470210902816461

Saxby, G. (Novemeber 19, 2010). The science of Imaging. In *The Science of Imaging*. CRC Press.

Scholte, H. S., Losch, M. M., Ramakrishnan, K., de Haan, E. H. F., & Bohte, S. M. (2018, January 1). Visual pathways from the perspective of cost functions and multi-task deep neural networks. *Cortex*. Where to Go Now with "What & How", *98*, 249–261. doi:10.1016/j.cortex.2017.09.019

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning* (pp. 1889–1897).

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015, June 8). High-Dimensional Continuous Control Using Generalized Advantage Estimation. arXiv: 1506.02438 [cs]. Retrieved July 8, 2018, from http://arxiv.org/abs/1506.02438

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., … Hassabis, D. (2017, October). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. doi:10.1038/nature24270

Sun, Y., Gomez, F., & Schmidhuber, J. (2011). Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *International Conference on Artificial General Intelligence* (pp. 41–51). Springer.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press. Retrieved from http://incompleteideas.net/book/RLbook2018trimmed.pdf

Tatler, B. W., Baddeley, R. J., & Gilchrist, I. D. (2005). Visual correlates of fixation selection: Effects of scale and time. *Vision research*, *45*(5), 643–659.

Tucker, G., Bhupatiraju, S., Gu, S., Turner, R. E., Ghahramani, Z., & Levine, S. (2018). The mirage of action-dependent baselines in reinforcement learning. *arXiv preprint arXiv:1802.10031*.

Uhlenbeck, G. E., & Ornstein, L. S. (1930, September 1). On the Theory of the Brownian Motion. *Physical Review*, *36*(5), 823–841. doi:10.1103/PhysRev.36.823

Ungerleider, L. G., & Haxby, J. V. (1994, January 1). 'What' and 'where' in the human brain. *Current Opinion in Neurobiology*, *4*(2), 157–165. doi:10.1016/0959-4388(94)90066-3

van Hasselt, H., & Wiering, M. A. (2007, April). Reinforcement Learning in Continuous Action Spaces. (pp. 272–279). doi:10.1109/ADPRL.2007.368199

Williams, R. J. (1992, May 1). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3-4), 229–256. doi:10.1007/BF00992696

Wolfe, J. M., & Horowitz, T. S. (2017). Five factors that guide attention in visual search. *Nature Human Behaviour*, *1*(3), 0058.

Yamins, D. L. K., & DiCarlo, J. J. (2016, March). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, *19*(3), 356–365. doi:10.1038/nn.4244

Yang, S. C.-H., Lengyel, M., & Wolpert, D. M. (2016). Active sensing in the categorization of visual patterns. *Elife*, *5*, e12215.

Yang, S. C.-H., Wolpert, D. M., & Lengyel, M. (2016). Theoretical perspectives on active sensing. *Current Opinion in Behavioral Sciences*, *11*, 100–108.

Appendices

A Validation Scores Translated EMNIST

Table A.1

All combinations Translated EMNIST

Algorithm	Context	Noise	Shared	Advantage	Score
Reinforce	Yes		Yes	State-Value	0.8508
PPO	Yes	Noise-Layer	Yes	State-Value	0.8482
PPO	No	Noise-Layer	Yes	State-Value	0.8460
PPO	No		Yes	State-Value	0.8448
PPO	No		Yes	Horizon-Aware	0.8448
PPO	Yes	Parameter	Yes	State-Value	0.8440
Reinforce	Yes		Yes	Horizon-Aware	0.8433
PPO	No	Noise-Layer	Yes	Horizon-Aware	0.8433
Reinforce	Yes	Noise-Layer	Yes	State-Value	0.8426
PPO	Yes	Noise-Layer	Yes	Horizon-Aware	0.8415
PPO	Yes		Yes	Horizon-Aware	0.8408
Reinforce	Yes	Noise-Layer	Yes	Horizon-Aware	0.8392
Reinforce	No		Yes	State-Value	0.8367
PPO	No	Parameter	Yes	State-Value	0.8356
PPO	Yes	Noise-Layer	No	State-Value	0.8355
Reinforce	No		Yes	Horizon-Aware	0.8308

Table A.1

All combinations Translated EMNIST (continued)

Algorithm	Context	Noise	Shared	Advantage	Score
PPO	No		No	Horizon-Aware	0.8288
Reinforce	No	Noise-Layer	Yes	State-Value	0.8285
PPO	Yes		No	Horizon-Aware	0.8273
Reinforce	No	Noise-Layer	Yes	Horizon-Aware	0.8268
PPO	No	Noise-Layer	No	Horizon-Aware	0.8260
PPO	Yes		No	State-Value	0.8247
PPO	No		No	State-Value	0.8222
PPO	No	Noise-Layer	No	State-Value	0.8218
PPO	Yes		Yes	State-Value	0.8136
PPO	Yes	Noise-Layer	No	Horizon-Aware	0.8076
Reinforce	Yes	Noise-Layer	No	Horizon-Aware	0.8073
Reinforce	No		No	State-Value	0.8021
PPO	No	Parameter	No	State-Value	0.7983
Reinforce	No	Noise-Layer	No	State-Value	0.7963
Reinforce	No	Noise-Layer	No	Horizon-Aware	0.7893
Reinforce	No		No	Horizon-Aware	0.7871
Reinforce	No	Parameter	No	State-Value	0.7685
PPO	Yes	Parameter	No	State-Value	0.7543
Reinforce	Yes	Parameter	No	State-Value	0.7523
Reinforce	Yes		No	State-Value	0.7408
Reinforce	Yes		No	Horizon-Aware	0.7225
PPO	Yes	Parameter	Yes	Horizon-Aware	0.4154

Table A.1

All combinations Translated EMNIST (continued)

Algorithm	Context	Noise	Shared	Advantage	Score
PPO	Yes	Parameter	No	Horizon-Aware	0.4008
Reinforce	Yes	Parameter	No	Horizon-Aware	0.3514
Reinforce	No	Parameter	Yes	State-Value	0.3390
Reinforce	Yes	Parameter	Yes	State-Value	0.3246
PPO	No	Parameter	Yes	Horizon-Aware	0.3243
PPO	No	Parameter	No	Horizon-Aware	0.2051
Reinforce	No	Parameter	Yes	Horizon-Aware	0.1300
Reinforce	No	Parameter	No	Horizon-Aware	0.1219
Reinforce	Yes	Parameter	Yes	Horizon-Aware	0.0602
Reinforce	Yes	Noise-Layer	No	State-Value	0.0579

B Validation Scores Cluttered EMNIST

Table B.1

All combinations Cluttered EMNIST

Algorithm	Context	Noise	Shared	Advantage	Score
PPO	Yes		No	State-Value	0.8284
PPO	Yes	Parameter	No	State-Value	0.8283
PPO	Yes	Noise-Layer	No	State-Value	0.8271
PPO	Yes		No	Horizon-Aware	0.8214
PPO	Yes	Noise-Layer	No	Horizon-Aware	0.8207
PPO	No		No	State-Value	0.8199
PPO	Yes	Noise-Layer	Yes	State-Value	0.8194
PPO	Yes	Noise-Layer	Yes	Horizon-Aware	0.8178
PPO	Yes		Yes	Horizon-Aware	0.8174
Reinforce	Yes		No	State-Value	0.8168
Reinforce	Yes	Parameter	No	State-Value	0.8140
PPO	Yes		Yes	State-Value	0.8138
PPO	No	Noise-Layer	No	State-Value	0.8108
Reinforce	Yes		No	Horizon-Aware	0.8052
Reinforce	Yes		Yes	State-Value	0.8032
PPO	Yes	Parameter	Yes	State-Value	0.8012

Table B.1

All combinations Cluttered EMNIST (continued)

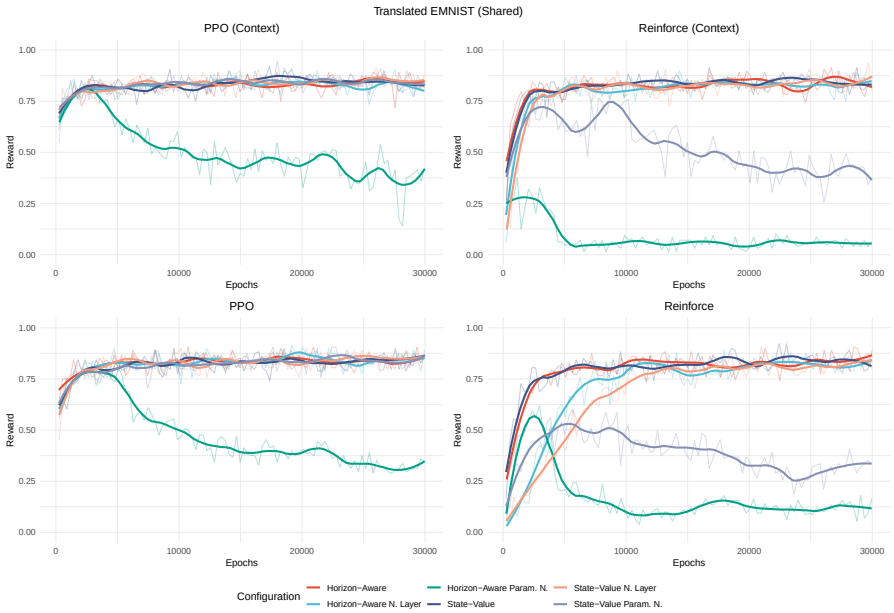
Algorithm	Context	Noise	Shared	Advantage	Score
Reinforce	Yes		Yes	Horizon-Aware	0.7962
Reinforce	Yes	Noise-Layer	Yes	Horizon-Aware	0.7936
Reinforce	Yes	Noise-Layer	Yes	State-Value	0.7850
PPO	No	Noise-Layer	Yes	State-Value	0.7839
PPO	No	Parameter	No	State-Value	0.7796
PPO	No		Yes	State-Value	0.7790
PPO	No	Noise-Layer	No	Horizon-Aware	0.7684
PPO	No		No	Horizon-Aware	0.7666
PPO	No	Noise-Layer	Yes	Horizon-Aware	0.7654
PPO	No		Yes	Horizon-Aware	0.7600
PPO	No	Parameter	Yes	State-Value	0.7448
Reinforce	No		Yes	State-Value	0.7448
Reinforce	No	Noise-Layer	No	State-Value	0.7234
Reinforce	No		Yes	Horizon-Aware	0.7218
Reinforce	No		No	State-Value	0.7122
Reinforce	No	Noise-Layer	Yes	State-Value	0.7053
Reinforce	No	Parameter	No	State-Value	0.6912
Reinforce	No		No	Horizon-Aware	0.6655
Reinforce	No	Noise-Layer	Yes	Horizon-Aware	0.6530
Reinforce	No	Noise-Layer	No	Horizon-Aware	0.6238
PPO	No	Parameter	Yes	Horizon-Aware	0.3348
PPO	Yes	Parameter	Yes	Horizon-Aware	0.3089

Table B.1

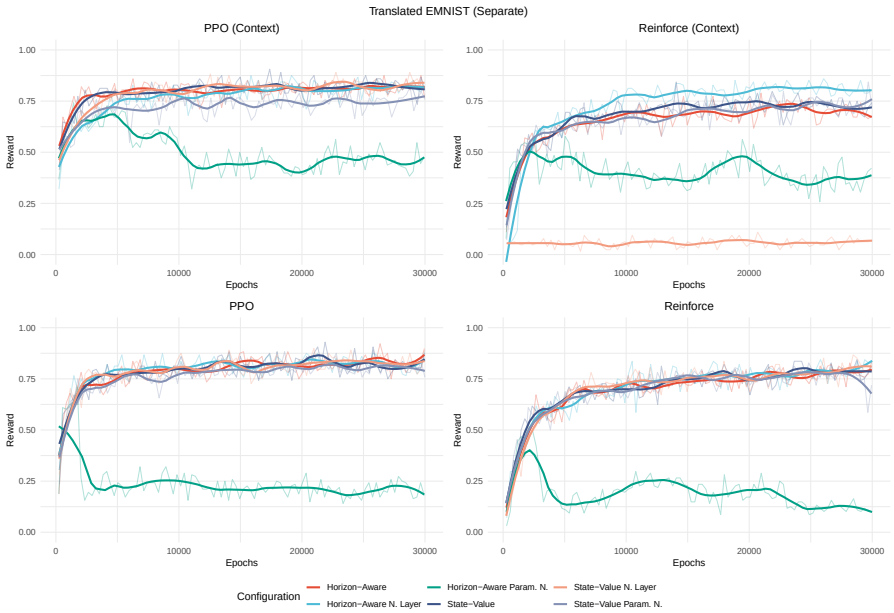
All combinations Cluttered EMNIST (continued)

Algorithm	Context	Noise	Shared	Advantage	Score
PPO	Yes	Parameter	No	Horizon-Aware	0.2619
Reinforce	No	Parameter	No	Horizon-Aware	0.0978
PPO	No	Parameter	No	Horizon-Aware	0.0679
Reinforce	No	Parameter	Yes	Horizon-Aware	0.0440
Reinforce	Yes	Parameter	No	Horizon-Aware	0.0396
Reinforce	Yes	Parameter	Yes	Horizon-Aware	0.0396
Reinforce	Yes	Noise-Layer	No	Horizon-Aware	0.0381
Reinforce	Yes	Noise-Layer	No	State-Value	0.0377
Reinforce	No	Parameter	Yes	State-Value	0.0343
Reinforce	Yes	Parameter	Yes	State-Value	0.0236

C Training Progress Translated EMNIST



(a) Shared parameters



(b) Separate parameters

Figure C.1. Development of validation reward during training on the Translated EMNIST task for all 48 configurations. Split on shared vs Separate Parameters and further divided by algorithm and context. Lines display the baseline used and the type of exploration noise (Parameter space, Noise layer or None).

D EMNIST Pixel distribution

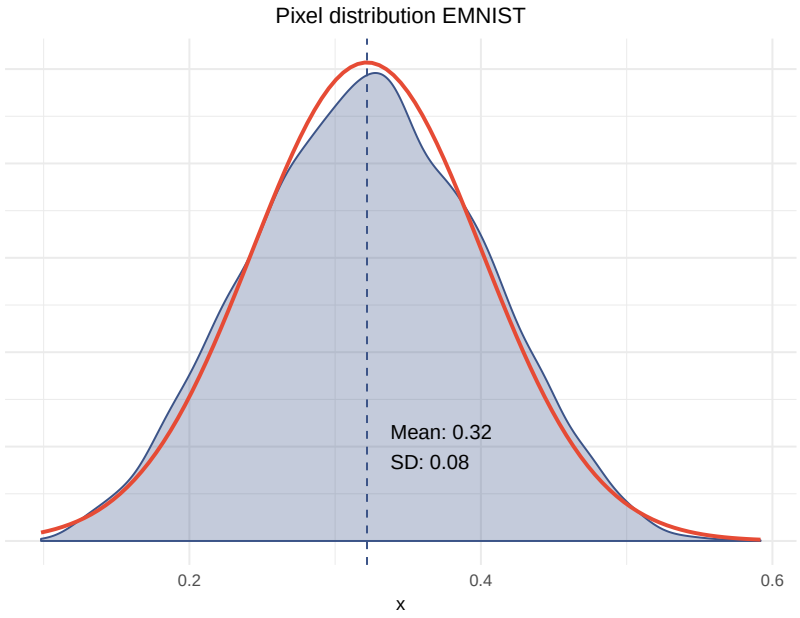


Figure D.1. Blue surface shows the relative distribution of pixels for the individual EMNIST characters. A normal distribution is fitted to the values as indicated by the red line and the Mean and SD values.