Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Bachelorarbeit Informatik

Florentin Doll

3. November 2025

Gutachter

Prof. Dr. Martin Butz
Kognitive Modellierung
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Prof. Dr. Nitram Ztub
Kognitive Modellierung
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Betreuer

Tomáš Daniš
Kognitive Modellierung
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Doll, Florentin:

Bachelorarbeit Informatik
Eberhard Karls Universität Tübingen
Bearbeitungszeitraum: 17.07.2025 – 17.11.2025

Abstract

Humans perceive the visual world through a process of selective attention, allowing them to focus high resolution vision on relevant parts of a scene and integrate that information across time. This thesis aims to model this process of visual attention with artificial agents, enabling them to efficiently process visual information in complex environments. We propose a novel (can i write novel here? it feels fancy but not sure) architecture that combines a recurrent visual attention mechanism with reinforcement learning to allow agents to learn where to look in order to maximize task performance. An auxiliary decoder helps to stabilize latent memory representations and enables interpretability of the current memory state. We evaluate the proposed architecture on a set of challenging visual tasks, including maze navigation, ...

Kurzfassung

Contents

Chapter 1

Introduction

Human vision is inherently selective. Rather than processing the entire visual field uniformly, the human eye relies on a foveated view, where high-resolution vision is concentrated in a small central region (the fovea), while the surrounding peripheral vision is of lower resolution. Humans sequentially direct their gaze towards interesting or task-relevant regions of a scene, integrating information over time to form a coherent understanding of their environment. This mechanism of selective attention allows humans to efficiently process complex visual scenes under limited computational resources.

In contrast, most artificial vision systems process visual inputs in a uniform manner, requiring high computational resources to achieve comparable performance to human vision. They also usually only use a single feedforward pass to process an image, leading to limited interpretability and adaptability to changing environments.

Research in active vision and attention-based reinforcement learning has begun to address these limitations. Models such as the Recurrent Attention Model (RAM) (Mnih *et al.*, 2014) and its variants have demonstrated the potential of foveated vision and sequential attention mechanisms in artificial agents.

Chapter 2

Methodology

2.1 Architecture

In this chapter, we describe the proposed architecture for modeling visual attention in artificial agents. Since the aim of the architecture is to mimic human visual attention, we need to incorporate several key components:

- **Encoder:** A visual input mechanism that allows the agent to take in information from the image.
- **Memory:** A memory module that enables the agent to maintain a memory of past visual inputs and integrate that information over time.
- **Agent:** A reinforcement learning framework that allows the agent to learn from its interactions with the environment and improve its attention and decision-making strategy over time.
- **Decoder:** As a final component, we introduce an auxiliary decoder that helps to stabilize the learning process and improve the interpretability of the agent's internal representations.

Together they make the agent be able to reason about where to look in a visual scene and how to use that information to perform tasks effectively in a similar way to humans.

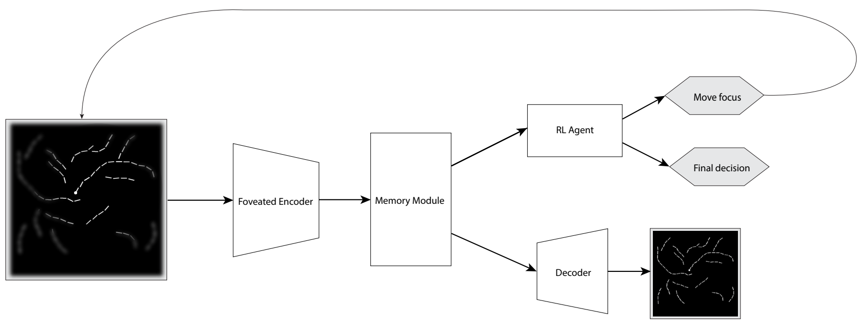

Figure 2.1: Network architecture diagram illustrating the key components of the proposed model.

2.1.1 Encoder

The encoder is responsible for processing the visual input and extracting relevant features. For the purpose of our architecture, we use a convolutional neural network (CNN) as the encoder and keep that one relatively small, since it will only process small cutouts of the image. We looked into and tested different ways of implementing foveated vision, such as multi-scale image patches, gaussian blurring or specific downsampling techniques. In the end we decided to go with a simple multi-scale patch extraction approach, similar to the one used in the Recurrent Attention Model (RAM) (Mnih *et al.*, 2014). That means we take multiple square patches of increasing size around the current gaze location, downsample them to a fixed size and pass all of those patches through the same CNN encoder. Using this method allows us to simulate foveation while keeping the encoder architecture and compute requirements simple and efficient.

2.1.2 Fusionlayer

After extracting features from the different patches, we need to combine them into a single feature representation. For that, we use a simple fully connected fusion layer that takes the concatenated features, aswell as encoded gaze coordinates as input and produces a fixed-size feature vector as output. We use it to preprocess the information into a vector that is easier to interpret and work with for the memory module. This Idea originates from the Glimpse Network from (Mnih *et al.*, 2014).

2.1.3 Memory Module

The memory module is a crucial component of the architecture, as it allows the agent to, step by step, build an understanding of the environment. For this we use a standard long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) as the memory module. The LSTM takes the fused feature vector from the encoder as input at each time step and updates its internal state accordingly. The hidden state of the LSTM serves as the agent's memory representation, which is used to feed into both the agent and the auxiliary decoder.

2.1.4 Agent

The agent is an actor-critic based reinforcement learning framework that consists of three policy heads and one value head:

- The first policy head outputs logits over the 8 possible discrete move actions (Categorical).
- The second policy head outputs a Bernoulli logit for the stop action terminate the episode.
- The third policy head outputs a binary classification logit for the final descision.
- The value head predicts $V(s_t)$ for advantage estimation.

All of these heads share a common fully connected layer that processes the LSTM hidden state, combined with the current gaze location before feeding it into the individual heads.

2.1.5 Decoder

The auxiliary decoder is designed to reconstruct the visual input based on the agent's memory representation. The purpose of the decoder is twofold: first, it encourages the memory module to learn meaningful and stable representations of the visual input. Training encoder, fusion layer and memory together with just the reinforcement learning signal from the agent would most likely lead to unstable training and poor performance. The reconstruction loss from the decoder provides an additional training signal that helps to stabilize the learning process and gives the latent space more structured representations and gradients. Second, the decoder allows us to interpret the agent's internal memory state by visualizing what the agent 'remembers' about the visual input at each time step. Do note that the decoder and agent, while being trained jointly, do not share any weights or parameters or loss signals. Therefore their interpretation of the latent space might be different. However, both components ultimately use the same memory representations and make decisions based on them so even though they might have different perspectives, the decoder still gives us valuable insights into the agent's understanding of the visual input.

2.2 Losses

The overall loss function for training the model consists of two main components: the reinforcement learning loss from the agent and the reconstruction loss from the auxiliary decoder. Both of these losses train the encoder, fusion layer and memory module jointly, while the agent only trains with the reinforcement learning loss and the decoder only trains with the reconstruction loss. The total loss can be expressed as:

$$\mathcal{L}_{total} = \mathcal{L}_{RL} + \mathcal{L}_{rec} \tag{2.1}$$

where $\mathcal{L}_{RL}$ is the reinforcement learning loss and $\mathcal{L}_{rec}$ is the reconstruction loss.

2.2.1 Reinforcement Learning Loss

For the reinforcement loss, we use GAE (Generalized Advantage Estimation) (Schulman *et al.*, 2018) to compute the policy gradient and update the agent's parameters. In the beginning we used A2C (Advantage Actor-Critic) (Mnih *et al.*, 2016) as the RL algorithm, but later switched to PPO (Proximal Policy Optimization) (Schulman *et al.*, 2017) since it generally provides more stable training and better performance.

We will not go into detail about the specific implementation of the RL loss here, as it is a well-established method in the field of reinforcement learning. The only thing that might not be standard is the way we handle the STOP action during training, since we have many components the early episodes are quite complex to learn, so we have to be careful of early stopping. Therefore we punish stopping early very heavily if the agent is either not confident about its decision or came to the wrong conclusion and stopped early.

2.2.2 Reconstruction Loss

The reconstruction loss is primarily based on the L1 loss between the original input image and the reconstructed image produced by the decoder. However, since the agent only sees parts of the image, we only compute the reconstruction loss over the areas that the agent has actually observed through its gaze, since the fully reconstruction isnt actually our goal. The goal of this loss is to stabilize the latent representations in the memory module, since we can assume that if we can reconstruct the seen parts of the image well, the latent representation must contain useful information about those parts. We dont care about the unseen parts of the image, since the agent has no information about them anyway. For more complicated to reconstruct datasets we used a combination of multiple losses on top of the L1 loss.

- **Perceptual Loss:** This loss is based on a pretrained VGG19 network (Pihlgren *et al.*, 2024) and helps to capture high-level features in the reconstructed images and therefore helps shaping the latent space. It computes the L2 loss between feature maps of the original and reconstructed images at different layers of the VGG19 network.
- **SSIM Loss:** The Structural Similarity Index Measure (SSIM) (Wang *et al.*, 2004) loss helps to improve the structural quality of the reconstructed images and encourages the model to focus on important features.
- **Gradient Discrepancy Loss:** The GDL loss (Mathieu *et al.*, 2016) focuses on matching the gradients between pixels of the original and reconstructed images, which helps to preserve edges and fine details in the reconstructions.

The final reconstruction loss is a weighted combination of all the mentioned losses at the last step of a reconstruction task, as well as a masked L1 loss at each step to actively encourage step-by-step reconstruction:

$$\mathcal{L}_{rec} = \lambda_{L1}\mathcal{L}_{L1}^{steps} + \lambda_{L1}^{final}\mathcal{L}_{L1}^{final} + \lambda_{perc}\mathcal{L}_{perc} + \lambda_{ssim}\mathcal{L}_{ssim} + \lambda_{gdl}\mathcal{L}_{gdl} \tag{2.2}$$

2.3 Pretraining

Since training all of the components at once from scratch is quite difficult and unstable, we use a two-stage training process. In the first stage, we pretrain the decoder alone, using an autoencoder setup. Through this we give the decoder an expected latent space 'structure', that the memory module and encoder can learn to adjust to. That structure will get overwritten or changed in the second training stage, but having a good starting point helps getting nice gradients from the reconstruction loss and therefore helps avoiding local minima. This is crucial since local minima are a big problem for complex architectures trained with reinforcement learning.

Chapter 3

Conclusion and Future Work

Appendix A

Important Additional Stuff

Bibliography

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

Mathieu, M., Couprie, C., and LeCun, Y. (2016). Deep multi-scale video prediction beyond mean square error.

Mnih, V., Heess, N., Graves, A., and Kavukcuoglu, K. (2014). Recurrent models of visual attention.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning.

Pihlgren, G. G., Nikolaidou, K., Chhipa, P. C., Abid, N., Saini, R., Sandin, F., and Liwicki, M. (2024). A systematic performance analysis of deep perceptual loss networks: Breaking transfer learning conventions.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2018). High-dimensional continuous control using generalized advantage estimation.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, **13**(4), 600–612.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum

Unterschrift