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

Bachelorarbeit Informatik

Florentin Doll

13. November 2025

Gutachter

Prof. Dr. Martin Butz
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 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 reasoning tasks, demonstrating its ability to learn effective attention policies. The aim of this thesis is to lay the groundwork for future research in active vision and reinforcement learning, providing a modular and extensible framework for exploring these ideas further.

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. In addition to these attention mechanisms, auxiliary tasks such as image reconstruction have been shown to improve the stability and performance of reinforcement learning agents (Jaderberg *et al.*, 2016).

In this thesis, we propose a novel architecture that combines a recurrent visual attention mechanism with an auxiliary reconstruction task to enable artificial agents to efficiently process visual information. Our agent will be trained to learn two policies, a decision making policy that classifies the input based on the glimpses seen so far, and a sensory policy that decides where to look next. This sensory policy will move the gaze to points of interest on the input image, integrating information over time to enable the decision making policy to make accurate classifications. We want to combine the benefits of active vision with the stabilizing effects of auxiliary tasks to create a robust and interpretable model for visual reasoning tasks.

This work will be a foundation that can be built upon and improved in future research and adapted to more difficult tasks, such as dynamic environments or real-world applications.

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.

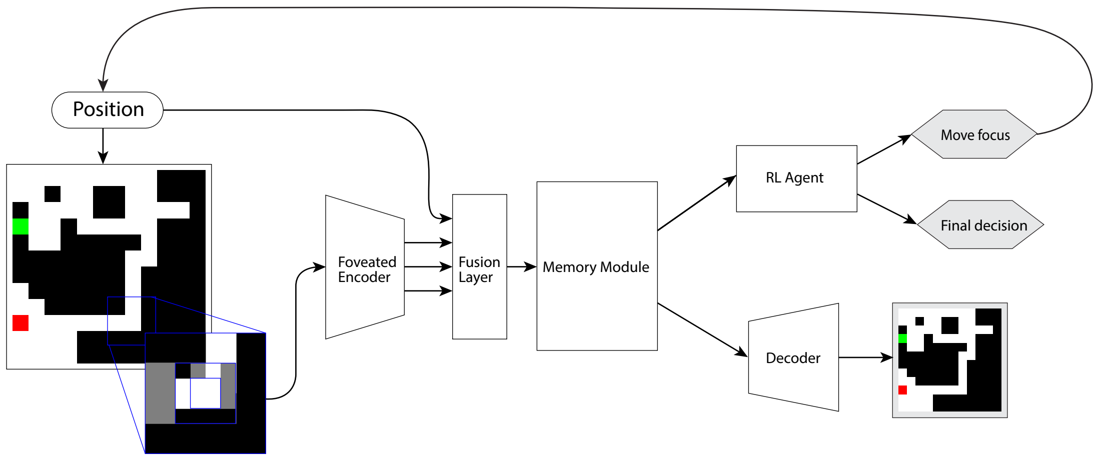


Figure 2.1: Network architecture diagram illustrating the key components of the proposed model.

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. We intended to later switch to PPO (Proximal Policy Optimization) (Schulman *et al.*, 2017) since it generally provides more stable training and better performance. However there seemed to be an issue with our PPO implementation that made our model train significantly worse, in heinsight that was not an issue with the PPO implementation but some other weird error, since even after reverting to A2C the performance was still worse than before. (This was fixed by going back to an older commit)

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. In practice that seemed to still not be enough since even though we couldnt observe early stopping in the early episodes, forcing the model to take at least a certain number of steps for the first half of the training seemed to help a lot with stability and performance.

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.

2.4 Datasets

Throughout this thesis we considered three main datasets for evaluating our architecture:

2.4.1 Compositional Visual Reasoning (CVR)

We initially planned to use the Compositional Visual Reasoning (CVR) dataset (Zerroug *et al.*, 2022) for evaluating our architecture. The CVR dataset consists of synthetic images, where there are always four images arranged in a 2×2 grid, containing different objects and shapes. The task is to find the odd one out based on a set of compositional rules. While this dataset is interesting and challenging, we figured other datasets would be more suitable for our specific architecture and research goals.

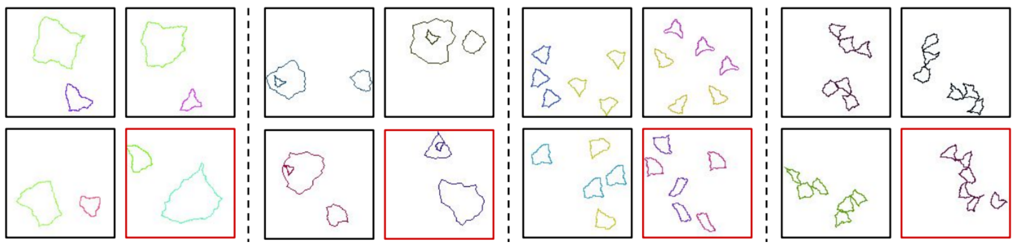


Figure 2.2: Example from the CVR dataset (Reproduced from (Zerroug *et al.*, 2022))

2.4.2 Pathfinder

The Pathfinder dataset (Tay *et al.*, 2020) is another dataset we considered for evaluating our architecture. It consists of images with two dots and a bunch of dashed lines, where the task is to determine whether there is a continuous path connecting the two dots. This dataset is very interesting for our architecture, since it requires long-range visual reasoning and attention to detail. However, it turned out that the very thin lines with most of the image being just background made it very difficult to reconstruct, which made training the auxiliary decoder challenging.

2.4.3 Maze Dataset

The Maze dataset is a new synthetic dataset we created, which consists of simple mazes with a clear path from start to end. The task is for the agent to navigate

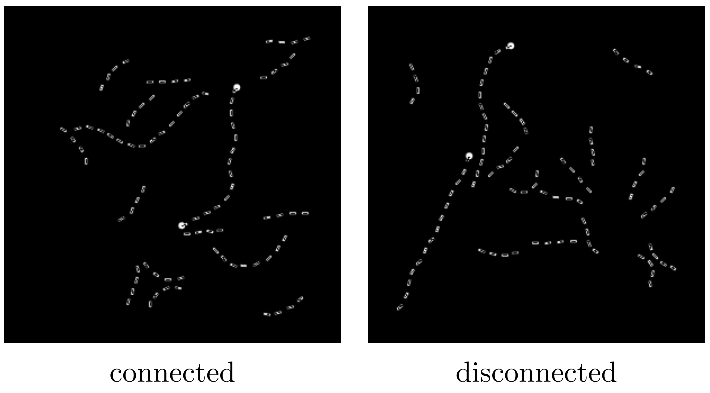


Figure 2.3: Pathfinder examples (Generated using (drewlinsley, 2019))

from the start to the end of the maze, while classifying whether such a path exists or not. Connected mazes carve a main path from start to end using depth-first search and then add random branches to increase complexity.

Mazes are built in a grid, where each cell has 4 pixels and is either a wall, a free space, or start/finish. These grids can be of any size, e.g. 10×10 cells (40×40 pixels) or 15×15 cells (60×60 pixels). We also have options to get specific path lengths, and can therefore control the complexity of the mazes.

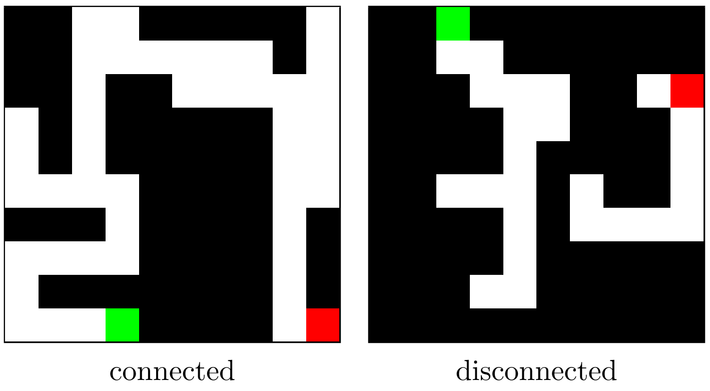


Figure 2.4: Maze examples (10×10 Grid)

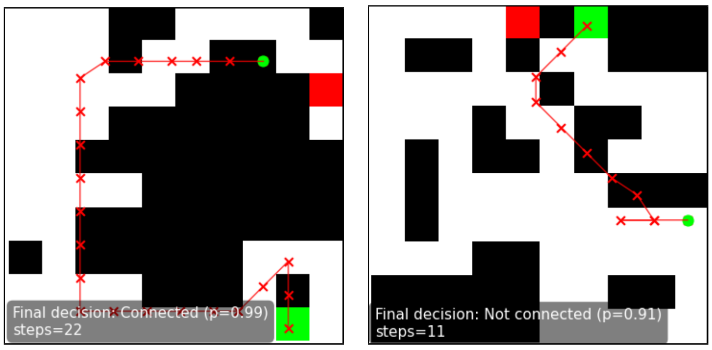
Chapter 3

Results

In this section, we report results for the proposed gaze-control model. On the Maze task we achieved promising overall performance. We can observe nice exploration behavior of the agent and fairly good generalization to unseen mazes.

3.1 Model Performance

With our current model we are able to achieve an accuracy of around 96.5% on the validation set, that consists of 3000 randomly generated mazes.


Final decision: Connected (p=0.99)
steps=22
Final decision: Not connected (p=0.91)
steps=11

connected disconnected

Figure 3.1: Randomly picked Validation Examples

These are some randomly picked validation examples of the maze task in Figure 3.1. As we can see the agent is able to successfully classify both the connected and disconnected maze. We can observe quite nice behavior of the agent, especially in the disconnected example, where the agent actually sees the goal in the very first glimpse and then goes on to try and find a path to it. Only after exploring all of the walls surrounding it and not finding a path to the goal it correctly classifies the maze as disconnected.

Also just to show how the agent perceives the maze we can look at Figure 3.2, which shows the last glimpse of the connected example in Figure 3.1. As we can see the agent doesnt get a lot of information on each step and has to rely on its memory to be able to successfully navigate the maze.

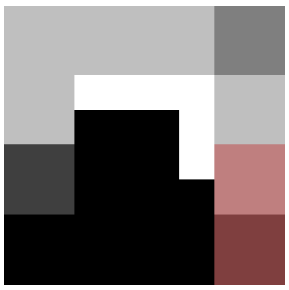
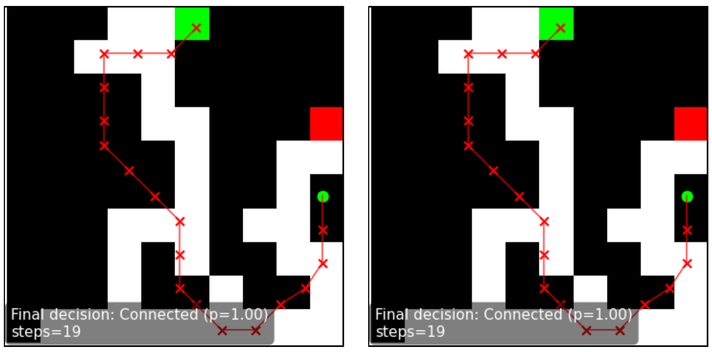


Figure 3.2: Glimpse at the last step of the exploration of the connected example in Figure 3.1

3.1.1 Exploration Behavior


Final decision: Connected (p=1.00)
steps=19
Final decision: Connected (p=1.00)
steps=19

original modified

Figure 3.3: Showcase of exploration behavior of the agent

In Figure 3.3 we took one of the validation mazes, where we saw it skipping over a dead end without looking into it and moved the goal into that deadend. As we can see the agent memorizes where it skipped a path and after not finding the goal in the other path it backtracks and explores the deadend it skipped before, showing that it is able to remember its previous observations and adapt its exploration strategy accordingly.

3.1.2 Starting Position

Since the starting position of the agent is chosen randomly on the start patch we can also analyze how much the starting position matters for the consistency of the agent. As shown in Figure 3.4 there are certain mazes where the starting position can have a big impact on the decision of the agent.

That issue is most likely caused by the combination of us using a discrete action space and the blurring of the outer areas of the glimpse. Because we only work with very little pixels per glimpse the blurring can either have a big impact on what the agent actually sees or not, depending on the exact position. If a wall is aligned with the blur, there wont be any blurring done at all, while if the wall is slightly off the blur will make it much harder to see.

In theory the agent should be able to realize that the information it has is uncertain and therefore explore more to be able to make a better descision in such cases. However that doesnt seem to always happen and therefore lead to these inconsistencies.

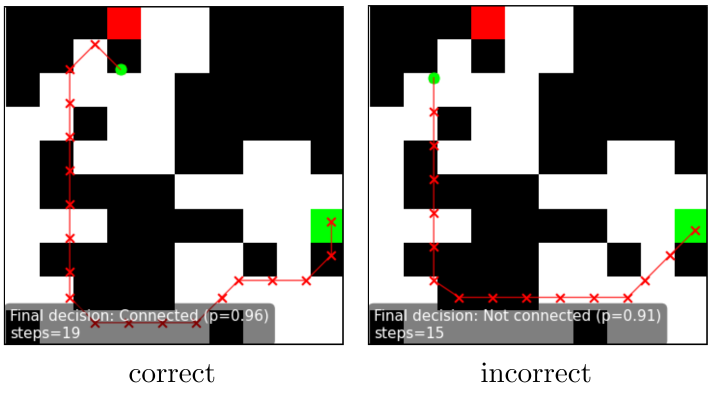

Final decision: Connected (p=0.96)
steps=19
Final decision: Not connected (p=0.91)
steps=15
correct
incorrect

Figure 3.4: An example for a maze where the initial position matters

3.1.3 Handpicked Mazes

To further analyze the performance of our model we also created a set of handpicked mazes that are meant to be more challenging for the agent, just to explore its capabilities further. These mazes are build to highlight certain challenges for the agent, like long corridors, lots of deadends or special structures that wont be found in randomly generated mazes.

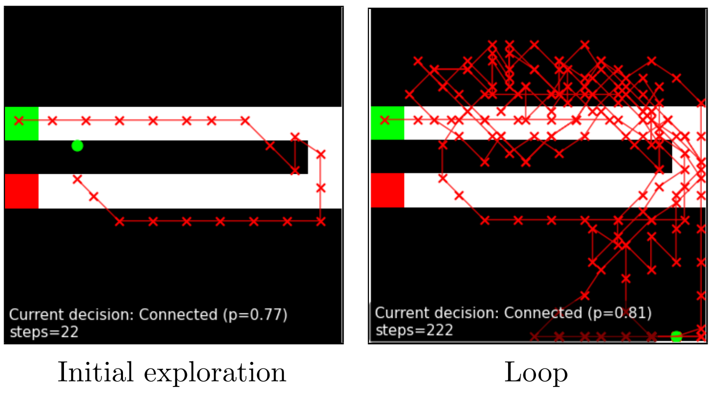

Current decision: Connected (p=0.77)
steps=22
Current decision: Connected (p=0.81)
steps=222
Initial exploration
Loop

Figure 3.5: Behavior on one of the handpicked mazes

In Figure 3.5 we can see one of these handpicked mazes, where the agent initially has a pretty decent run. It explores the entire corridor and finds the goal. However the decision head doesnt get certain enough about the connectivity of the maze and therefore doesnt stop the run. After that the agent aimlessly wanders around, getting stuck in a loop.

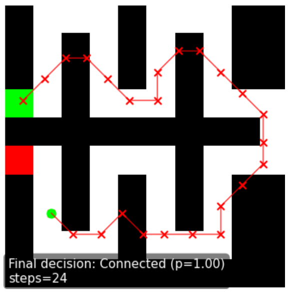


Figure 3.6: Solving of a pretty complicated maze

Another interesting example is shown in Figure 3.6, where the agent is able to successfully navigate a pretty complicated maze structure. However if we slightly modify the maze structure to make the wiggles a bit larger the agent fails to properly make a descision and ends up looping again, as shown in Figure 3.7. The exploration itself looks solid, but it doesnt reach a certain enough confidence to make the descision to stop. Then we can oberve a quite nice backtrack over the path, back to the start but even after that it still cant reach a certain enough confidence to stop the run.

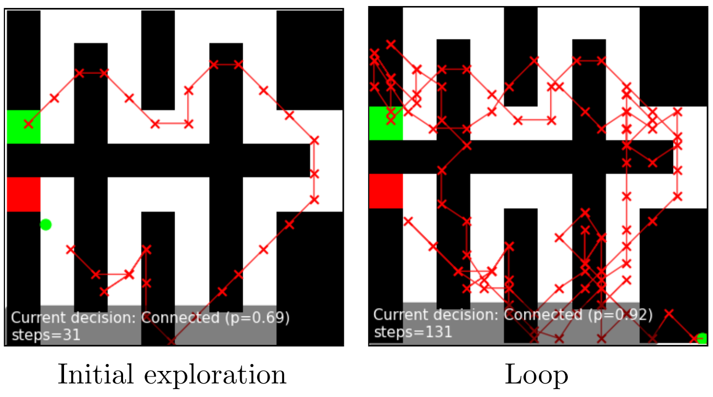

Current decision: Connected (p=0.69)
steps=31
Current decision: Connected (p=0.92)
steps=131
Initial exploration
Loop

Figure 3.7: Slightly modified to increase wiggle size

Also, without showing images of all the examles here, we found that the agent generally struggles with more simple structures, like for example a single line of walls with one gap, that seperates start and goal or a diagonal wall without any gaps. These structures are are very different from the random mazes the agent was trained on and therefore it struggles to properly explore them and make a descision.

We also found that the more complex the mazes get, the more the starting position matters for the final descision of the agent, as mentioned in Subsection 3.1.2.

There will be some results on different maze sizes here to show generalization, aswell as maybe some runs with different hyperparameters

3.2 CNN Classifier comparison

To have a comparison we also trained a baseline model that was just a simple CNN classifier. For this task a baseline will get almost perfect results for randomly generated mazes as it can easily learn to recognize the goal pattern in the input image. If we use similar amounts of parameters for the baseline model as for our gaze-control model we can observe an accuracy of over 99.50%, while the gaze-control model achieves around 96.5% accuracy.

However this comparison is not entirely fair since the classifier has access to the full image at all times, while our gaze-control model only gets a very limited view of the maze. So while we do run into efficiency problems with our model, since it has a lot of parameters and needs multiple steps to explore the maze, we use way less information to make the descision. The classifier gets the full mazes of 40×40 pixels, so 1600 pixels in total, while our gaze-control model only gets 3 glimpses scaled to 4×4 pixels each, so only 48 pixels in total for each glimpse. So as long as we need less than 33 glimpses (for reference, during validation we averaged below 13 steps per maze) we use less information than the classifier to make a descision.

There will be stats on the impact of bigger maze sizes here

To further dive into comparing capabilities of the models we also tested them on the handpicked mazes from Subsection 3.1.3. These mazes are build to highlight the capabilities of both models.

This will also be added

Lastly we also created a set of mazes that the classifier model fails on, while excluding trivial cases like the start being right next to the goal since the classifier for some reason often failed on those. Without these the set was composed of pretty difficult and long mazes that required a lot of exploration to find the goal. On these mazes the gaze-control model was able to achieve a success rate of around 60%, showing that it definitely can have an advantage over a simple classifier in certain scenarios, even in fairly simple environments like these mazes.

3.3 Ablation Studies

To better understand the contributions of different components of a model its common to test the model without certain components. For our gaze-control model we will always need a certain core structure to be able to perform the task, which includes the encoder, the LSTM and the agent, since removing these would defeat the point of the architecture. However we can ablate some of the additional components that we added to the model to see how much they contribute to the overall performance.

3.3.1 Decoder

In particular we tested the model without the decoder branch and therefore without the auxiliary loss.

The results are not fully done yet, they will be added

Without the auxiliary loss the training progress was significantly slower and reached a lower final performance. Surprisingly though the model actually reached a fairly decent performance even without the auxiliary loss, showing that the reinforcement learning signal can actually be enough to train the model to a certain degree. However this is only tested on this fairly simple maze task and that actually makes sense. The auxiliary loss helps to stabilize training by helping the lstm memorize what was seen. If the task is simple enough the model can get away with just using the reinforcement learning signal to learn a decent exploration strategy.

We expect that on more complex datasets the auxiliary loss will be even more crucial for training since the sparse reinforcement learning signal will most likely not be enough to train the entirety of the model.

3.3.2 Fusionlayer

The Fusionlayer was meant to basically preprocess the information of the 3 glimpses and the location into a single vector of information so that the LSTM is only responsible for memorization.

This ablation study would have been interesting to see how much the Fusionlayer actually helps with training the model. However due to time constraints we decided against running this experiment since this was also an idea taken from the RAM paper (Mnih *et al.*, 2014) and therefore is at least somewhat validated.

Chapter 4

Limitations

Along the way of developing and testing our model we ran into some limitations that we want to discuss in this chapter.

4.1 Reconstruction of Datasets

As mentioned in Chapter 2 we originally wanted to use the Pathfinder and CVR datasets to evaluate our model on more complex data. However due to the way we implemented our auxiliary loss, using a decoder to reconstruct the input image we ran into a lot of problems since these datasets are not really made to reconstruct and therefore are super hard to reconstruct for the model. They consist of thin lines on a mostly blank (or white for the CVR dataset) background which gives the model a huge local minima that is very hard to avoid.

We tried various different loss functions but they were either not enforcing a strong enough learning signal to avoid the local minima or we couldnt use them for local losses around the glimpse and could only run them on the full image, which also harms training since the model never saw some of the regions that it gets penalized on.

We are certain that these problems can be fixed. Be it either with a better way of training the decoder or with a different auxiliary task that is better suited for these datasets. One of these different auxiliary losses is for example the Sensorimotor Reward from SUGARL (Shang and Ryoo, 2023) where they basically punish gazes that do not contribute useful information, which is measured in the accuracy of the decision making policy. However that was not within the scope of this thesis and therefore we had to limit ourselves to the simpler maze dataset.

4.2 Parameter and Training Efficiency

On more simple tasks like the maze dataset our model is pretty inefficient in terms of parameters and training time compared to the classifier model. Due to the neccessity of using an LSTM that keeps track of the seen glimpses our model has a lot of parameters compared to other models that just use feedforward architectures.

Additionaly the training time is significantly higher since we have a lot of components that need to be trained together and the reinforcement learning through time steps is a lot more noisy than a simple classification loss.

However what we lack in efficiency we gain in flexibility, since our model could for example easily adapt to differently sized mazes without retraining or changing the architecture at all (except the decoder, which is not needed for classification though). Another major advantage is that our model is already setup to work in a dynamic environment, since we already integrate information over time steps and can therefore directly be used in a setting where the input is changing over time, like a video stream from a robot navigating in an environment or a game.

Chapter 5

Conclusion and Future Work

In this thesis we presented an architecture for using active vision reinforcement learning models to solve visual reasioning tasks.

5.1 Summary of Findings

- We proposed a gaze-control architecture that combines an encoder–policy module with a decoder branch for auxiliary reconstruction loss.
- We evaluated the proposed gaze-control architecture on our Maze benchmark and were able to show that our model is able to successfully learn to classify mazes by actively exploring them with a limited field of view.
- Though on average worse than a feedforward classifier with full image access, our model showed promising results and was able to solve mazes that the classifier failed on.
- By using ablation studies we were able to show that the decoder branch did improve the training stability and convergence speed of the model.
- We put a lot of effort into trying to make our model work on more complex datasets like the Pathfinder and CVR datasets, however we ran into limitations regarding the reconstruction of these datasets.

The main contribution of this thesis is not necessarily the performance of the model on the Maze dataset, but rather the groundwork that we laid for future research in this area. We provided a reliable and modular implementation of the gaze-control architecture along with a reproducible data pipeline and evaluation scripts. This foundation can be used to easily extend the model to more complex datasets and objectives in future work.

5.2 Future Work

There are a lof of interesting directions for future work that build upon the groundwork laid in this thesis.

- **Different Datasets:** The most obvious next step is to extend the model to more complex datasets like the Pathfinder and CVR datasets. This would require addressing the limitations regarding the reconstruction of these datasets that we discussed in Section 4.1. But we are confident that this can be achieved with some modifications to the auxiliary loss or the decoder architecture. It might also be interesting to see what happens if we just accept the reconstruction to be fully black, since we are mainly interested in the loss signal and not in the actual reconstruction.

- **Alternative Auxiliary Tasks:** Another interesting direction is to explore alternative auxiliary tasks that are better suited for complex datasets. For example, the mentioned Sensorimotor Reward from SUGARL (Shang and Ryoo, 2023) or predictive coding could be used as auxiliary tasks instead of reconstruction.

- **Transformer-based visual sequencing:** Explore replacing the recurrent state with a transformer encoder that ingests sequences of foveated patches with explicit positional / temporal embeddings, possibly enabling improved modeling of spatial / temporal dependencies.

- **Transfer Learning:** The modular nature of the model makes it well-suited for transfer learning scenarios. Future work could investigate how well the model can transfer knowledge from one dataset to another, or how well it can adapt to new tasks with limited data.

- **Dynamic Environments:** As earlier mentioned, our model is already setup to work in dynamic environments. This would be an exciting direction for future work, where the model could be tested in scenarios where the input is changing over time, like a video stream from a robot navigating in an environment or a game. Robots are a really hot topic right now and combining active vision with robotics could lead to some really interesting applications.

Appendix A

Important Additional Stuff

Bibliography

drewlinsley (2019). Pathfinder. `https://github.com/drewlinsley/pathfinder`. Accessed: 2025-11-07.

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

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks.

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.

Shang, J. and Ryoo, M. S. (2023). Active vision reinforcement learning under limited visual observability.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. (2020). Long range arena: A benchmark for efficient transformers.

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.

Zerroug, A., Vaishnav, M., Colin, J., Musslick, S., and Serre, T. (2022). A benchmark for compositional visual reasoning.

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