UNIVERSITY OF LEEDS

Bio-inspired Reinforcement Learning: Algorithm Development and its Application to Visual Search

Zhile Yang

Submitted in accordance with the requirements for the degree of Doctor of Philosophy

The University of Leeds

Faculty of Engineering & Physical Sciences

School of Computer Science

July 2025

Intellectual Property and Publication Statements

I confirm that the work submitted is my own, except where work which has formed part of jointly authored publications has been included. My contribution and the other authors to this work has been explicitly indicated below. I confirm that appropriate credit has been given within the thesis where reference has been made to the work of others.

In this work, chapters 2, 3, and 4 include work that has been published in a jointly-authored publication. The details of the publication are as follows:

Z. Yang, S. Guo, Y. Fang, Z. Yu and J. K. Liu, "Spiking Variational Policy Gradient for Brain Inspired Reinforcement Learning," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 47, no. 3, pp. 1975-1990, March 2025, doi: 10.1109/TPAMI.2024.3511936.

I completed the theoretical derivations, experiments, visualizations and analyses of the results, and the text and the appendix.

Shangqi Guo, Ying Fang, Zhaofei Yu, and Jian K. Liu contributed to the formulation of the ideas in the theoretical analysis and the design of the experiments. Shangqi Guo, Ying Fang and, Jian K. Liu contributed to the drawing of the sketches of the network, that is, Figure 1 and 2 in the publication.

Signed 杨知乐

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Drs. Jian Liu, Yongxing Wang, David Head, and my other thesis group members, Drs. Yanlong Huang, He Wang, and Arash Rabbani for their guidance and support. A special thought goes to Dr. Jian Liu. The countless meetings and suggestions are indispensable for my academic development.

I would like to thank my collaborators on the SVPG project. Their ideas and contributions are precious to the outcome of the project.

I would like to thank all my colleagues at the Neural Computation Laboratory. Their inspirations and experiences helped me overcome challenges in research and life.

I would like to thank my family for their invaluable support. Despite the large physical distance, their help is always available and is the one that helps me get through the hardest times.

This project made use of: ARC3 and Aire, part of the high performance computing facilities at the University of Leeds; BlueBEAR and Baskerville, part of the high performance computing facilities at the University of Birmingham; and facilities of the N8 Centre of Excellence in Computationally Intensive Research (N8 CIR) provided and funded by the N8 research partnership and EPSRC (Grant No. EP/T022167/1). The Centre is co-ordinated by the Universities of Durham, Manchester and York. Baskerville was funded by the EPSRC and UKRI through the World Class Labs scheme (EP/T022221/1) and the Digital Research Infrastructure programme (EP/W032244/1) and is operated by Advanced Research Computing at the University of Birmingham.

Abstract

The field of reinforcement learning has seen significant advances in recent years. However, there are still many challenges, including adaptability to environmental changes, robustness to noise, energy efficiency, safety, etc. A promising direction is to incorporate neuroscience findings to explore the potential of replicating the strong cognitive abilities of humans and animals, which, in return, can also contribute to our understanding of brain functions.

In this work, I propose a new model of spiking neural network and derive a reinforcement learning algorithm for it. The algorithm is based on reward-modulated spike-timing-dependent plasticity, thus having better biological plausibility. Experiments on standard reinforcement learning tasks demonstrate its ability to solve challenging tasks and have better inherent robustness to a variety of perturbations than standard methods.

My method is also applied to real-life visual search scanpath modeling tasks that are more challenging. Additionally, I design a new map-based inverse reinforcement learning method that can better extract motivations from scanpaths. Experiments show the effectiveness of the spiking neural network in solving the scanpath modeling task. To obtain an in-depth understanding of the cognitive mechanisms of visual search behaviors, I further apply the reinforcement learning method to the analysis of scanpath properties of social and non-social behaviors of visual search. The results offer new understandings of the patterns of eye movements.

Taken together, the results presented in this thesis provide novel insights into not only developing new reinforcement learning algorithms but also understanding the behaviors and mechanisms of our visual search function.

Contents

List of Figures

List of Tables

Abbreviations

ANN	artificial neural network
ANN2SNN	ANN-to-SNN
ASD	autism spectrum disorder
BP	backpropagation
CNN	convolutional neural network
DNN	deep neural network
DQN	deep Q-network
DRL	deep reinforcement learning
GAIL	generative adversarial imitation learning
GPU	graphics processing unit
IBS	ideal Bayesian searcher
IOR	inhibition of return
IRL	inverse reinforcement learning
LIF	leaky integrate-and-fire
LTD	long-term depression
LTP	long-term potentiation
LSTM	long short-term memory
MDP	Markov decision process

MLP	multi-layer perceptron
MME	map-based maximum entropy
MSE	mean squared error
PPO	proximal policy optimization
RL	reinforcement learning
RWTA	recurrent winner-take-all
R-STDP	reward-modulated spike-timing-dependent plasticity
SNN	spiking neural network
SRM	spike response model
STDP	spike-timing-dependent plasticity
SVM	support vector machine
SVPG	spiking variational policy gradient
WTA	winner-take-all

Chapter 1

Introduction

1.1 Background

Reinforcement learning (RL), as a third type of machine learning after supervised learning and unsupervised learning, models the interaction process of an artificial agent and its environment [1]. This enables it to solve sequential control tasks. Rather than using pre-collected datasets, RL collects data automatically in the interaction process, so it does not rely on an expert's understanding of the task or the design of datasets.

A core design of RL methods is the mapping from state observations to state values or a policy distribution. In classic methods such as Q-learning [2], a look-up table is used to represent the mapping. This type of representation has a memory cost that grows exponentially with the dimension of the state space and action space. This hinders those methods from being applied to tasks with high-dimensional state space or action space, including vision-based control tasks and tasks with a continuous action space, which are prevalent in real-world control tasks. Deep reinforcement learning (DRL) methods, originating from the deep Q-network [3], use a neural network to present the mapping. With the optimization method which is based on backpropagation (BP), and the development in network structures such as convolutional layers [4], long short-term memory (LSTM) [5], and Transformer [6], neural networks can approximate a wide range of mappings from and to high-dimensional spaces. The potential ability in generalization could also accelerate learning on similar inputs. These benefits enable RL to solve tasks with high state and action dimensions, including video games [7, 8], robot control [9], chess [10], and machine translation [11].

Despite the above successful applications, RL agents are still far from human intelligence. Most RL applications are still limited to simulated [7, 8] or in-laboratory scenes [9, 12, 13] where trial-and-error is acceptable. For out-of-laboratory applications, the tasks generally need to be simplified through abstracted states or action representations [14, 15]. As mentioned in a review [16], the low sample efficiency of training and poor adaptability of policies are two of the core problems of RL. In addition, the high energy consumption of commonly used devices, e.g., graphics processing units (GPUs), is also a problem, especially for real-world applications with limited energy supply [17].

For these problems, many existing studies revise RL algorithms from perspectives including reward shaping [18], task abstraction [19], task augmentation [20–22], policy amendment [23,24], and parallelization [25], etc. These studies generally use conventional neural networks, and the improvements are achieved through amendments to policy structure, environment model, or computing paradigms. Considering that current performances are still not ideal when compared to humans, and that conventional neural networks are only a coarse approximation of the human brain, it is promising to consider more inspiration from the brain in the design of RL methods.

Brain-inspired RL studies incorporate neuroscience findings into the designs of neural networks to explore replicating the cognitive abilities of human brains. The neuron models, network structures, and learning methods are designed to more precisely simulate human brains. On one hand, it has been found that these networks can improve the transfer performance [26] and energy efficiency [27] of the learning system. On the other hand, brain-inspired RL studies also contribute to the analysis and understanding of brain mechanisms such as biological neuron signals [28] and cognitive functions [29]. An important type of cognitive behavior is visual search, which reflects the mechanism for visual attention of the brain [30]. In the below sub-sections, the background of brain-inspired RL methods and the visual search modeling are introduced respectively.

1.1.1 Brain-Inspired RL Methods

A mainstream of brain-inspired RL methods uses spiking neural networks (SNNs) to represent functions of values or policies. SNNs differ from conventional artificial neural networks (ANNs) primarily in their spiking neuron models, which model the activities as discrete spikes. Such neuron models better resemble brain neurons, but also make the gradients of the network parameters unavailable. This necessitates special designs for training an SNN.

SNN RL methods can be categorized into three types: ANN-to-SNN (ANN2SNN) [31], surrogate gradient function [32, 33], and reward-modulated spike-timing-dependent plasticity (R-STDP) [34, 35]. The first two types require backpropagation of gradients during training and are not considered biologically plausible [36–38], meaning that they are not considered to be hypothetically realizable by biological brains. In contrast, R-STDP methods train with local learning rules and thus are considered biologically plausible [38–40]. In addition, R-STDP is also preferable for implementation by neuromorphic hardware [41].

Recent studies investigated different forms of R-STDP, considering types of neuronal interactions, modulation strengths, and modulation timings [34], and have seen improvements in task performance and energy efficiency [42–44]. However, most existing methods are constrained to shallow networks with only one hidden layer, and the applications are limited to simple tasks [39, 42, 45, 46]. A key issue is that the local learning rules in R-STDP may not correctly represent the RL task target, i.e., there is a gap between them. A recent work [47] proposed a solution for bridging the gap based on variational inference. Nevertheless, the method is still only derived for a three-layer network and tested on simple tasks.

1.1.2 Visual Search

Visual search is a kind of cognitive behavior when humans search for a certain target given a search image. Understanding the visual search behavior provides insights into brain mechanisms and benefits applications to robot vision, autonomous driving, design of human-computer interface, and mental healthcare [30, 48, 49]. The behavior is generally captured as sequences of fixation coordinates with reference to the search image, also known as scanpaths. The dependence on stimulus images and the randomness of scanpaths make them challenging to model. One of the main topics in visual search is behavior prediction, where predictive models are built to predict the human scanpaths given the same search and target image as humans.

Existing methods for scanpath prediction can be categorized according to their types of machine learning, i.e., unsupervised [50, 51], supervised [49, 52], RL [53, 54], and inverse reinforcement learning (IRL) [55–57]. Among them, IRL methods have an advantage in explicit extraction of the motivation behind the behaviors. This can be achieved through learning a reward function that drives the agent to behave similarly to humans. The reward function can serve as a more integrated and less noisy representation of scanpaths for behavior analysis.

Nevertheless, there is a problem with existing IRL studies for scanpath modeling. The reward functions learned by existing methods are dependent on the agent policy [55,57–60]. Therefore, a static representation of the motivation with reference to the search images cannot be obtained. This hinders the comparison of the impacts of different parts of the search image on human visual search behavior, which is important to mechanism analysis.

Another issue with existing scanpath modeling studies is the biological plausibility of the models. A biological model better facilitates the analysis of how brains work to produce cognitive behaviors. However, most existing methods use conventional ANNs for the prediction task. Although a recent study adopted SNNs [61] to build brain-inspired models for this cognition task, the model in this study is trained by backpropagation, which is not considered biologically plausible.

1.2 Motivations

The motivation of this thesis is to deepen our understanding of the fundamental mechanisms of the human brain by developing novel brain-inspired RL methods. In particular, it focuses on visual search, which is a sequential and interactive cognitive process that reflects the brain's mechanism for visual attention. This can advance the construction of brain models, as well as help detect cognition-related disorders such as autism spectrum disorder (ASD).

Biologically plausible computational methods, which respect biological constraints on neural structure and learning, can establish a connection between neural architectures and behavioral functions. Visual search, as a form of interactive cognitive behavior, can therefore be naturally investigated using brain-inspired RL approaches.

However, developing biologically plausible RL methods that are capable of addressing complex tasks, such as visual search, remains challenging. Meanwhile, existing visual search models struggle to extract a static motivation representation that is suitable for mechanism analysis. Therefore, to advance this area, we need a biologically plausible RL method that scales to complex tasks like visual search, along with a technique that supports explicit motivation extraction.

1.3 Objectives

To address the above research gaps, this thesis sets up the following objectives:

- Design a new biologically plausible RL method that improves the performance of existing methods and solves challenging tasks such as video games. The performances are evaluated on standard benchmark RL tasks. The indicator for solving tasks is to achieve comparable scores to standard methods.
- Design a new visual search modeling method that supports extracting static motivation representations. The model should achieve comparable performances in behavior prediction to standard methods on standard visual search datasets.
- Evaluate the proposed methods and models on the different types of visual search tasks, aiming to extract the underlying brain mechanisms in various behavioral patterns.

1.4 Contributions

To achieve the above objectives, I proposed recurrent winner-take-all (RWTA), a new network structure for SNN. The network consists of recurrently-connected winner-take-all (WTA) circuits [62, 63]. Adopting the mean-field variational inference method, which has been demonstrated to be effective in transforming global target into local learning rules [47, 64], I demonstrated that the fixed point of the network is equivalent to an energy-based RL policy formulation, which has a good capability in representing complex mappings. Based on this, I proposed a last-step approximation to derive the R-STDP learning rules, named SVPG, that optimize the RWTA network according to the RL policy gradient. SVPG was also extended to more popular RL algorithms such as proximal policy gradient (PPO) [65].

I systematically evaluated SVPG over five typical RL tasks, including reward-based MNIST classification [66], Gym InvertedPendulum [67], ViZDoom HealthGathering [68], AI2THOR robot navigation [69, 70], and robot arm manipulation [71]. Among them, the ViZDoom task, as a 3D first-person video game, is the most challenging as it involves image input and long decision sequences. The robot tasks use near photo-realistic scenes [69, 71] and randomized starting/target positions, which examine the method's applicability to real-world tasks. Empirical results show that SVPG can solve all five tasks and outperforms the compared R-STDP method. SVPG also outperforms three representative methods, including ANN, ANN2SNN, and surrogate gradient-based backpropagation in terms of optimization speed. Furthermore, SVPG exhibits inherent robustness to input noise [31], network parameter noise [72], and environmental variation [73].

For visual search, I proposed a map-based maximum entropy (MME) IRL method that decouples the reward function into a static, policy-independent reward map and a set of stimulus-independent environmental reward functions. MME can extract a reward map that can be directly used as a motivation representation. Experiments on visual search datasets verify the effectiveness of MME in replicating human scanpaths. SVPG was further applied to MME, and experiments demonstrate its capability of solving the scanpath prediction task. In addition, MME and two standard behavior cloning methods were applied to an ASD dataset [30] to investigate the behavior patterns of social and non-social stimuli. Previous results on dissimilarity between social and non-social patterns were based on simple statistics [30]. My findings are consistent with those results, while offering new understandings uniquely offered by my proposed predictive models. My research introduces a new paradigm for the interpretation of visual scanpaths through predictive modeling.

The main contributions of this thesis are summarized as follows.

- A new SNN structure is proposed and the corresponding R-STDP learning method is derived. This new method improves the capability of R-STDP methods to solve challenging RL tasks.
- A new IRL method is proposed that enables the extraction of a policy-invariant motivation representation for visual search tasks.
- The proposed R-STDP method is applied to the IRL visual search task. For the first time, a biologically plausible agent is built for predicting human visual search behavior. This paves the way for visual search behavior understanding with biologically plausible systems.
- The proposed IRL method is applied to the analysis of behavior patterns of social and non-social stimuli and provides new insights into potential mechanisms of brain disorders.

1.5 Structure of the Thesis

The remainder of this thesis is organized as follows.

Chapter 2 provides an overview of RL models. It reviews the mainstream standard approaches for RL and brain-inspired approaches for RL methods, highlighting their relationship and connections between different types of methods. It then describes the visual search behaviors task and the relevant existing studies on using RL for understanding visual search behaviors.

Chapter 3 proposes SVPG, a new type of biologically plausible neural network model, and its corresponding implementation of the RL method. It presents the theoretical derivation for SVPG and discusses additional designs of the model for practical considerations in real tasks.

Chapter 4 presents the evaluation of SVPG on standard RL tasks and compares it to other representative models. It presents the empirical verification of assumptions made in the theory part, then shows the performance on different benchmark RL tasks as well as the results of the perturbation tests, ablation tests, and visualizations.

Chapter 5 proposes MME, a new IRL method, to better understand the motivation behind visual search, which can extract policy-independent reward maps. It presents results on applying the MME with SVPG and other compared models to standard visual search datasets, and a more in-depth analysis of social and non-social behaviors when ASD is presented in subjects. The outcome of this chapter provides some new insights into understanding the underlying visual search mechanisms of the human brain.

Chapter 6 concludes this thesis by summarizing its key contributions, examining its limitations, and outlining promising directions for future research.

Chapter 2

Literature Review

This thesis develops a new brain-inspired SNN model and a new R-STDP learning method to improve the task performance of R-STDP methods on RL tasks. The proposed R-STDP method is applied to the modeling of visual search, a type of brain cognitive behavior, for potential in building the relationship between brain structures and cognitive functions as well as disease analysis such as ASD.

This chapter first provides an overview of RL methods, including mainstream standard approaches and brain-inspired ones. It then describes the visual search behavior modeling task and the existing studies on using RL and IRL methods for behavior prediction and on disease analysis. Along with the descriptions, preliminary formulations and notations are also introduced.

This chapter includes work that has been published in a jointly-authored publication [74]. Shangqi Guo, Ying Fang, and Jian K. Liu contributed to the drawing of the sketch of the WTA circuit, that is, Figure 1 in the publication, which corresponds to Figure 2.1 in this chapter.

2.1 Reinforcement Learning

RL models the interaction between an artificial agent and its environment. The interaction process is commonly modeled by the Markov decision process (MDP). In this thesis, I adopt the notations from [1] and use a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ to denote the elements in an MDP, which are respectively the state space, action space, state-transition function, reward function, and reward discount factor.

Considering that most real-world applications, e.g., robot control, autonomous vehicles, have a finite length of the interaction process, in this thesis, I focus on RL formulation with an indefinite time horizon, where the agent interacts with the environment in the form of episodes. In each episode, there is a maximum number of time steps. At each time step t, the agent observes the state $s_t \in \mathcal{S}$, makes action $a_t \in \mathcal{A}$ according to its policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ and receives a scalar reward r_t. The environment transfers to a new state s_{t+1} according to the transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. The learning objective is to find a policy π that maximizes the expected return:

$$J(\pi) = \mathbb{E}_{\tau \sim P_\pi} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right], \tag{2.1}$$

where P_π denotes the trajectory distribution over policy π, τ is the sampled trajectory $\langle s_0, a_0, r_0, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T \rangle$. Here T is the total length of the episode and could be either a fixed number or a variable, depending on tasks.

Depending on the formulation of the policy function π, RL methods can be divided into value-based and policy gradient methods. As an example of value-based methods, Q-learning [2] learns a value function $Q(s, a)$ and selects the action with the maximum value at each state s. In contrast, policy gradient methods directly optimize a policy distribution. An advantage of policy gradient is its flexibility in policy formulation, as it supports stochastic policies and continuous action representations. REINFORCE [75] is a classic policy gradient method. In this thesis, the derivation of the new R-STDP learning method is based on REINFORCE to utilize this advantage.

A key design in RL is the representation of the policy mapping. Classic methods typically use table [2] and linear regression [76, 77], which have a poor capability in representing complex mappings, thus limiting their applications to tasks with high-dimensional state or action spaces. In more recent studies, the introduction of deep neural networks (DNN) and backpropagation-based optimizer [3] replaced the previous choices of function approximators. This type of RL study is often referred to as DRL. The expressive power provided by DNN enables representing complex mappings, thus scaling up RL to solve challenging tasks, including video games [7], chess [10], and vision-based robot control [13]. Nevertheless, DNNs are not perfect. On one hand, the adoption of DNNs brings problems. For example, the values estimated by a neural network can be noisy and inaccurate, resulting in overestimation [78] and making training unstable [7]. An update to the network may not always improve the policy [79]. The replay buffer, often used in DRL, may reduce the learning efficiency when set to an improper size [80]. The lack of explainability of DNN hinders DRL from being applied to scenarios such as finance and medicine, where the decisions need to be explained [81]. On the other hand, there are still many scenarios where introducing DNN is not sufficient to solve the problem. The learning speed, adaptability to similar tasks, reliability in a real-world environment, and energy efficiency are far from human-level.

For these challenges, recent studies propose techniques from many different perspectives to improve the performance of RL methods. For example, reward engineering tunes the formulation of the reward function to improve sample efficiency [18, 19, 82]. Policy abstraction reduces the state or action dimension to reduce the complexity of the RL task [7, 83–85], to improve sample efficiency [86], and to improve adaptability to task changes [12, 87]. Task augmentation enlarges the set of training environments to improve the adaptability of the policy [88, 89]. Distributed calculation revises the computing paradigm of policy inference and optimization to improve their speed [25, 90]. Many of these techniques require task-specific designs [18, 82, 88]. This thesis proposes a new SNN for policy representation. It can be categorized as another type of improvement that is made to the policy function approximators. This type of improvement does not require task-specific knowledge, thus having better flexibility in applications.

2.2 Brain-Inspired Reinforcement Learning

Most of the DRL studies use conventional ANNs in the approximation of value or policy functions [7–9, 25]. In contrast, it is well-known that humans can solve complex real-world tasks, learn rapidly, and have good adaptability to new environments. Human brains also consume less energy than commonly used computing devices [17]. Considering the large difference between ANNs and human brains, it is possible that, through mimicking brains, some functions of humans can be replicated in artificial agents.

There are various ways to take inspiration from the brain. From the perspective of state representation, hyperdimensional computing has been used to encode the input observation into a high-dimensional space in a way that mimics human brains [91]. From the perspective of functional building blocks, compositional models with each component functioning like a specific brain area (e.g., medial prefrontal cortex, orbitofrontal cortex) have been proposed [92, 93]. This thesis takes another perspective that considers neuron models that better resemble the behavior of biological neurons. This perspective is more fundamental and task-independent, thus being more flexible for applications. A commonly used type of neuron model is spiking neurons, which take timing into firing dynamics and communicate through spikes [94]. The resulting network model is then referred to as SNN. This thesis focuses on SNN for RL. In the following sections, I will first review general designs of an SNN and then review studies on SNN for RL.

Because the true mechanism of the brain is unknown, the brain-inspired designs are based on existing neuroscience findings. An important feature of brain-inspired models is their biological plausibility, which means the extent to which a model or learning method can be realized by biological brains [95]. Following major studies, this thesis adopts spiking neurons [37, 63] and local learning rules [96–98] as the criteria. Specifically, in this thesis,

- A network of neurons is referred to as biologically plausible if the neurons communicate through discrete signals.
- A learning method is referred to as biologically plausible if the update to learnable parameters is based on local spike events instead of backpropagation of remote signals.

2.2.1 Spiking Neural Networks

SNN is a popular direction of brain-inspired models. Many researchers recognize SNN as the next generation of neural networks [97,99,100]. The design of SNNs mainly consists of four parts: spiking neuron models, information encoding, network topology, and learning methods [37].

Spiking neuron models. Early spiking neuron models, such as the Hodgkin-Huxley model [101], come from measurements of biological neurons and involve complex computation. The Izhikevich model [102] provides simplification while keeping most of the computational features [103]. Nevertheless, more simplified models like leaky integrate-and-fire (LIF) and spike response model (SRM) [104] neurons, although not as similar to biological neurons, are more commonly used in recent studies [37, 105, 106]. SRM is a stochastic variant of the LIF model [45], and has been widely used in RL studies [45, 105, 107, 108]. In this thesis, I also adopt SRM as the basis unit for the proposed network.

In SRM, the states of neurons evolve at discrete spike time steps. At each spike time step l, each neuron evolves its membrane potential $u(l)$ and fires a spike at random according to its firing probability $\rho(l)$. All spikes are considered to be binary, i.e., 1 for firing and 0 for resting. The firing probability $\rho(l)$ depends exponentially on the membrane potential [105], as defined in the following equation:

$$\rho(l) = \exp\{u(l) - I(l)\}, \tag{2.2}$$

where $u(l)$ is the membrane potential and $I(l)$ is an inhibitory term. In this thesis, it is designed to be produced by a lateral inhibition neuron in a WTA circuit, as will be introduced later. The membrane potential $u(l)$ is determined by the spike train S from connected neurons [109]:

$$u(l) = b + \sum_{j \in N(\cdot)} w_j \int_0^\infty \kappa(y) S_j(l - y) \mathrm{d}y, \tag{2.3}$$

where $N(\cdot)$ denotes the set of presynaptic neurons of the considered neuron, w_j is the synapse weight, κ is the excitatory postsynaptic potential, S_j is the spike train from neuron j, and b is the intrinsic excitability of the neuron. For κ, I consider the rectangle function due to its simplicity:

$$\kappa(l) = \kappa_0[\varepsilon(l) - \varepsilon(l - \tau_1)], \tag{2.4}$$

where κ_0 is the overall amplitude, τ_1 is a hyperparameter that determines the size of the time window of spikes that take effect, and $\varepsilon(l)$ is the step function. Future work may consider the double exponential function, which is more common in the literature [110, 111]. The double exponential function is given by:

$$\kappa(l) = \kappa_0[\exp(-l/\tau_2) - \exp(-l/\tau_3)]\varepsilon(l), \tag{2.5}$$

where τ_2, and τ_3 are hyperparameters.

Information encoding. This is the way the spike trains of neurons represent information. Basic coding methods include temporal coding and rate coding [37]. Temporal coding keeps the timing of spikes. An example is the time to first spike [37]. Rate coding uses the frequency of spikes over a time period to represent information. This is more like traditional ANNs but may not capture all the information in the spike trains because the timing of spikes is discarded [37]. Nevertheless, rate coding is widely used in SNN RL [112–114]. This thesis adopts the rate coding method for information encoding. There are also other coding methods. For example, population coding considers the firing activities of a group of neurons instead of a single one, and has been shown to increase the representation capacity of a network [113]. Future work

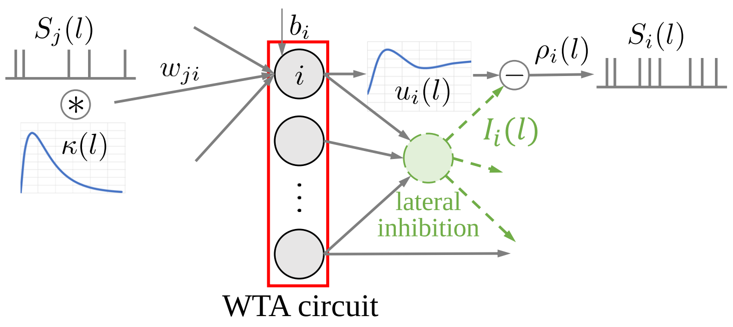


Figure 2.1: The illustration of the structure of a WTA circuit. The neurons are shown in a red box. Each neuron receives input spike trains (denoted as S_j) and changes its membrane potential u_i. The lateral inhibition neuron modulates the firing probability ρ_i. The output spike train S_i is sampled according to ρ_i.

may investigate the integration of different encoding methods with the network proposed in this thesis.

Network topology. The topology of a neural network is crucial for the representation capacity. Typical topologies of SNNs are consistent with traditional ANNs and include fully connected, recurrent, and convolutional structures [115]. More sophisticated structures like residual connections [116] and attention modules [117] have also attracted attention. Beyond these common structures, some studies focus on liquid state machines [118] and WTA circuits [63, 105]. Previous studies have demonstrated the capability of WTA circuits in implementing the inference of a hidden Markov model [62] and hierarchical Bayesian inference [63]. This thesis adopts the WTA circuit as a building block of the proposed network to utilize its potential capability in implementing policy inference.

The design of the WTA circuit in this thesis follows the idea from [63] and [47]. Figure 2.1 illustrates a WTA circuit of SRM neurons. There is a lateral inhibition neuron in the circuit that produces the inhibitory term $I(l)$ for the SRM neurons, as mentioned in Eq. (2.2). This inhibitory term is assumed to make the firing probabilities within the WTA circuit sum up to $\hat{\rho}$, resulting in a low firing rate of the whole network and low energy consumption. When $\hat{\rho} = 1$, this indicates that one and only one neuron in the WTA circuit fires at each time step.

Like traditional ANNs, the structure of an SNN can also evolve during training. Genetic algorithms [119] and growing-pruning [120] have been studied to find better topologies in training. Future work may investigate this direction to further improve the network structure.

2.2.2 Spiking Neural Networks for Reinforcement Learning

Most recent RL studies use a function approximator to represent a value function [7] or a policy distribution [65, 121]. SNNs for RL are used to replace the conventional ANN as the function approximator [31, 33, 114]. Conventional DRL methods use backpropagation of gradients to optimize the network [7, 25, 122]. Due to the fact that the output of a spiking neuron is non-differentiable, the backpropagation of gradient is infeasible, and special learning methods are necessary to apply SNNs to RL tasks. The learning methods can be categorized into three types: conversion, surrogate gradient, and R-STDP.

Conversion from ANN to SNN. ANN2SNN methods first train a traditional ANN and then convert the parameters to the target SNN. A series of studies has investigated the conversion of different building blocks of ANNs, including recurrent layers, softmax activation, max-pooling, and batch-normalization, and has gained comparative performances to original ANNs [123–125]. ANN2SNN allows the implementation of well-performing ANNs with SNNs and neuromorphic hardware to potentially improve energy efficiency. A drawback, however, is that the accuracy could be worse after conversion. To alleviate this problem, a long simulation time could be necessary, which results in high inference latency.

Surrogate gradient methods. Surrogate gradient methods use a differentiable function to approximate the derivative of the activation function. This enables the transfer of the gradient across the SNN and across spike time steps to update the parameters. Typical surrogate gradient functions include the rectangular function, triangular function, exponential, and the derivative of the sigmoid function [126, 127]. I refer readers to Figure 3 of [126] for an illustration of the surrogate gradient functions. Li et al. [127] also propose a series of hyperbolic tangent functions evolving the surrogate function during training. Note that the temporal dimension of SNN can cause a computational burden in the backpropagation of the gradient through time, especially when the simulation length is long. This is considered by the field of credit assignment, and various methods like e-prop [32] and feedback alignment [128] have been proposed to help alleviate the problem. An advantage of surrogate gradient methods over ANN2SNN methods is that they do not suffer from the conversion accuracy problem, so the simulation time can be reduced, leading to a shorter inference latency and higher energy efficiency. Nevertheless, both these types of methods rely on backpropagation of the gradient, which is not realizable by biological brains.

R-STDP methods. R-STDP is a modulated variant of spike-timing dependent plasticity (STDP), which is a framework that contains many of the learning rules identified by neuroscientists [97]. STDP is a variant of Hebbian learning rules, which state that the connection between two neurons is strengthened when the neurons fire in a causal manner [37]. STDP extends this by taking the specific timings of the firing activities into consideration [129]. Depending on the sign of the time interval between the postsynaptic and presynaptic spikes, the synaptic weight is either potentiated or depressed. This kind of weight update only depends on local signals and does not take into consideration a global target. R-STDP extends STDP by modulating the update to synaptic weight with an external signal, which is the reward signal in RL. This is supported by neuroscience findings about the relationship between dopamine, acetylcholine, and synapse plasticity [130, 131], and has been used in various RL studies on biologically plausible methods [45, 46, 107]. Because R-STDP method is local and does not require backpropagation, it is often commented to be more biologically plausible than the other two types of methods. Besides, R-STDP also does not require conversion from ANN, thus it does not suffer from the inference latency problem.

To introduce the formulation of R-STDP, I use w_{ij} to denote the weight of the synapse between presynaptic neuron i and postsynaptic neuron j. R-STDP takes the form of $\Delta w_{ij} = R(l) \cdot \mathrm{STDP}(l)$, where $R(l)$ is the external reward signal and $\mathrm{STDP}(l)$ is a coefficient determined by the STDP learning rule in the following form:

$$\begin{aligned}\mathrm{STDP}(l) = & S_j(l)\left[W_{\mathrm{pre}} + \int_0^{\infty} A_+ W_+(y) S_i(l-y)\mathrm{d}y\right] \\ & + S_i(l)\left[W_{\mathrm{post}} + \int_0^{\infty} A_- W_-(y) S_j(l-y)\mathrm{d}y\right],\end{aligned} \tag{2.6}$$

where W_{pre} and W_{post} are constants about the presynaptic and postsynaptic activity, A_+ and A_- characterize the extent to which synaptic changes depend on the current synapse weights. W_+ and W_- are respectively the time windows of the long-term potentiation (LTP) and the long-term depression (LTD) processes, which satisfy $\int_0^{\infty} W(l)\mathrm{d}l = 1$. The tuple $\langle W_{\mathrm{pre}}$, W_{post}, $A_+(w_{ij})$, $A_-(w_{ij})\rangle$ defines a specific STDP rule.

Starting from the original R-STDP [107], different variations of the modulation have been proposed, including TD-STDP [45], feedback-modulated TD-STDP [46], R-max [34], and hybrid [35]. Based on such methods, tasks including 2D goal-reaching [45], CartPole [45–47], LunarLander [46], and dynamic vision sensor-based lane keeping [39] have been solved. Nevertheless, these tasks are simpler than the ones that surrogate gradient-based and ANN2SNN-based methods can solve, such as Atari games. One deficiency of these R-STDP methods is that the neuron models and STDP rules are not adapted to RL algorithms and network structures. There is a gap between the local learning rules of R-STDP and the global RL task target. In fact, most listed studies use a shallow network structure with less than three layers [39, 45, 46]. This deficiency in the capability of representing complex mappings hinders them from being applied to more challenging RL tasks. Although there are methods such as BP-STDP [132] and using R-STDP for initialization [133] that scale up R-STDP to more complex networks, their algorithms introduce backpropagation operations and break biological plausibility. In contrast, variational inference, as adopted in previous studies [47, 64], is more promising as it approximately builds the relationship between a policy distribution and the SNN without using backpropagation. This thesis also adopts this idea. The differences to existing studies are that: study [47] only considers a simple structures only one hidden layer of WTA circuits, while this thesis considers a recurrent fully-connected structure of WTA circuits that is more general; the method in study [64] is designed for supervised learning tasks and does not implement the optimization with R-STDP, while this thesis focuses on RL tasks and R-STDP-based optimization.

The advantages of using SNNs in RL over conventional ANNs include energy efficiency, biological plausibility, as well as robustness to perturbations. In Atari video games, conversion from ANN to SNN brings better robustness to input occlusions [31]. A directly trained deep spiking Q-network exhibits better robustness to white-box attacks than a vanilla deep Q-network (DQN) [134]. For implementation with neuromorphic hardware, the noise in synaptic weights has been studied, and a certain type of SNN converted from an ANN is shown to be more robust than the original ANN in the MNIST classification task [72]. For energy consumption, an SNN-based policy implemented with Intel Loihi was shown to consume 75 times less electricity in a robot navigation task than a conventional ANN implemented on Jetson TX2 [112]. This thesis also offers a test of robustness where perturbations including input noise, network weight noise, and environmental variations are tested. Similar to existing studies, I also found that my SNN can obtain robustness to these perturbations, which supports the current understanding of the advantages of SNNs. The test of energy consumption requires neuromorphic hardware and is a promising direction of future work.

2.3 Visual Search

Visual search is the field that concerns the eye movements of humans when they are searching for something. There are two main topics in this field. One topic concerns the recognition and measurement of eye movements. This includes gaze tracking based on different types of observations like face images [135], videos [136], and observations from specialized devices [137]. Such studies provide a basis for developing tools for collecting eye movement data in the real world. Another main topic is the collection and analysis of the data. This typically involves designing behavioral tasks for human participants, data gathering, and data analysis. The data gathered are often called "scanpaths", which are sequences of fixations during a search attempt. Through analyzing the behavior patterns from scanpaths, such studies can offer insights about certain diseases such as ASD [30] or even help with the diagnosis process [138]. The work in this thesis belongs to the second topic. I use existing visual search datasets collected by other studies and focus on analyzing the behavior patterns. Below, I will first provide a brief introduction to datasets, types of analysis, and metrics, and then review existing methods for scanpath modeling.

Visual search datasets. Visual search datasets are collected when human participants perform a certain task. Typical visual search tasks require participants to search for a target image in a search image. Depending on the existence of the target image in the search image, the tasks can be described as target-present, target-absent, or a mixture of the two. For the search images, some studies use natural images [139, 140] and some use synthesized images [30, 50, 141]. For the target images, the form of presentation can be a category name [140], an exact portion from the search image [142], or a variable example image in the target category [50]. The procedure of data collection often varies. In [140], the participants are required to indicate whether the target image exists in the search image. In [141], a two-alternative forced choice paradigm is used. In [30], an extra step of clicking the target image with a mouse is included when the participant reports the existence of the target. Different designs of the human task and the stimuli can affect the behavior of human participants. Following some recent studies [50, 55], this thesis uses multiple datasets with different types of stimuli to more comprehensively evaluate the performance of the methods compared.

The type of human task and stimuli not only has effects on human behavior patterns but also determines the applicability of some predictive models. For example, fixation prediction methods based on similarities between the target image and patches of the search image [50, 51] may not be suitable for target-absent tasks or tasks where targets are represented with abstract labels. In general, a method designed for target-absent tasks can be easily adapted to target-present tasks, but the inverse is difficult. This thesis considers task-present settings so that more existing methods can be used for comparison.

I would like to note that although this work is concerned with visual search, the analysis of human eye movements is not limited to this type of behavior. There are also studies on reading [143,144], simulated driving [145, 146], etc. In addition, the process of gaining visual information is not limited to eye movements, but also head movements [53, 147]. The methods proposed in this work may also be helpful for other types of behavior.

Types of analysis. The visual search studies that analyze eye movement data can have various objectives. Two of the mainstream objectives are to (1) extract behavior patterns of subjects, and (2) build predictive models to replicate human behaviors.

Pattern extraction studies often use eye movement datasets collected from different types of subjects or task settings. A basic and commonly used approach is based on statistics of the movement data, such as the proportion of fixations in a certain area [148], fixation duration on areas of interest [149], and behavioral performance [30]. These indicators have been used in investigations of general behavior patterns [148], attractive regions of stimuli [149], and differences in social attention between people with and without ASD [30]. A more sophisticated method summarizes multiple scanpaths into one path [150] and uses the similarities between paths to detect autism. In [151], a support vector machine (SVM) is built to classify subjects with and without ASD based on statistical features of fixations. Deep learning models have also been investigated. In [152], an LSTM model is built to predict whether the scanpaths are obtained from a target-present, target-absent, or free-viewing setting. In [138], a model based on pre-trained saliency prediction is trained to predict whether a subject has ASD or not. A problem with this type of analysis is that the mechanism for generating the visual search behavior is not extracted. Existing findings may suggest certain patterns of the behavior, but the reason why the behavior is generated is unclear.

Studies on predictive models focus on understanding the mechanism that generates eye movements. The main objective is to build a model that, given the same stimuli as human participants, predicts eye movements similar to those of humans under certain criteria. Many early works take the saliency map as the target [48]. Saliency maps merge all fixations into a map and lose the sequential information. Recent works often aim to predict the scanpaths directly so that the sequential information in eye movements is preserved [49, 50, 55]. A general scanpath prediction model predicts a sequence of fixations given the same stimuli as humans. Some studies additionally predict the termination of the sequence [49, 52, 59, 152, 153], and the duration of fixations [52, 154]. This thesis also builds scanpath prediction models. Considering that some datasets do not offer duration information, I choose to neglect the prediction of durations. In addition, to use scanpath similarity metrics that do not support scanpaths with different lengths, I force the agent scanpath to have the same length as humans, which removes the need for termination prediction. Future work may consider adding the termination and duration to prediction targets. Below, I will present a detailed review of methods for scanpath prediction.

2.3.1 Methods for Scanpath Prediction

The methods for predicting scanpaths can be categorized into unsupervised learning, supervised learning, RL and IRL. Different types of methods bring different mechanisms for replicating human behavior. Note that some studies introduced below are beyond visual search and consider scanpaths collected from other types of tasks, e.g., free-viewing. They are included because a simple adaptation is possible to apply them to visual search. In the following, I will first review unsupervised learning, supervised learning, and RL methods. I will then elaborate on the formulation and methods of IRL.

Unsupervised learning methods

Unsupervised methods typically use heuristic rules to generate the scanpath. Examples of simple rules include random uniform distribution and center bias, as mentioned in a benchmark [142]. In IVSN [50], a pre-trained model is adopted to extract features from the images, the features are convolved to generate a saliency map, and a rule of inhibition of return (IOR) is applied to the saliency map to generate the scanpath. EccNET [51] extends this method to have dynamic saliency maps at each fixation step. In these two methods, the saliency map is built on similarities between patches of the search image and the target image. In [155], an image reconstruction model is built, and the reconstruction error is used to build the saliency model. In ideal Bayesian searcher (IBS) [156], a prior distribution and a visibility map are incorporated into the search process. Correlation IBS and structural similarity IBS [157] extend it to natural images by adopting pre-trained neural networks for the prior distribution and adding correlation or structural similarity to the estimation of the posterior distribution. A further extension, nnIBS, which changes the template similarity to be based on IVSN, is described in [142]. An advantage of these unsupervised learning methods is the transparency of the mechanism for scanpath generation. A disadvantage is that the heuristic rules or pre-trained saliency and detection models can have a limited range of applications.

Supervised learning methods

Supervised learning methods build models that learn from datasets, thus generally having a better fitting performance. Since the stimuli are images and the scanpath involves sequential prediction, a general model consists of convolutional neural networks (CNNs) for image processing and LSTM layers to capture sequential information. Sun et al. [154] build a probabilistic model based on CNN and LSTM to predict fixations. The log-likelihood of the ground truth fixation is used to train the model, along with auxiliary tasks including IOR map prediction and saliency map prediction. PathGAN [153] also adopts a CNN and some LSTM layers, but uses a combination of mean squared error (MSE) loss and adversarial loss by a discriminator to train the model. Beyond LSTM, in human attention Transformer [49], a Transformer-based encoder is designed to maintain an explicit working memory that captures history information. The memory is then used in a Transformer decoder to predict pixel-level fixations in a step-wise manner. There are also methods that do not rely on recurrent network components. In DeepGazeIII [158], the recent history of fixations is represented by a stack of coordinates of recent fixations and is treated as one source of input to the network. In Gazeformer [52], a Transformer is trained to output all the fixation predictions in a scanpath at once. Another feature of Gazeformer is the adoption of a language model to encode the semantic target description, which could help with generalization.

Reinforcement learning methods

A problem with the above-mentioned supervised learning methods is that the training objective may not be properly designed. An MSE loss or log-likelihood target specified by a Gaussian distribution means that fixations closer to each other are similar. This may not be suitable because of the potential noise in human behavior and different semantic information at fixation points. As will be introduced later in section 2.3.2, the similarity between a human scanpath and an agent scanpath is often measured by ScanMatch and MultiMatch, which are not differentiable and thus not directly applicable as the training objective for supervised learning. In contrast, RL methods construct an environment where the agent explores different scanpaths. The agent's scanpaths are evaluated by the reward function. Since the reward function does not need to be differentiable, RL methods support directly optimizing towards similarity measurements. As an example, in [54], the reward function is defined to be a combination of ScanMatch scores. In addition, since RL accumulates the reward in the learning objective, it builds the relationship between fixations in a sequence. In [53], a hand-crafted single-step reward function is proposed to measure the similarity of a single fixation, and a discounted sum of rewards is used as the training objective. Another advantage of RL is its interactive nature, which allows direct simulation of the human task. In [159], an RL agent is trained to perform saccades and to decide whether the target exists in the search image, which is the same as humans. The reward signal is generated to present the correctness of the judgment.

Inverse reinforcement learning methods

Despite the advantages of RL, a problem is that the motivation of the behavior cannot be explicitly extracted. A representation of motivation should be able to explain the agent's behavior, thus enabling comparison of the effects of different stimuli on human behavior. For this deficiency, a recent trend of studies uses inverse reinforcement learning (IRL) to learn the motivation representation and the behavior model simultaneously.

Formulation of general IRL. The formulation of IRL is similar to that of RL. An MDP $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ is used to formulate the interaction between the agent and the environment. Different from RL, the reward function R is not available and needs to be learned. In addition, some demonstration trajectories from an expert are available as $\mathcal{D} = \{\xi_1, \ldots, \xi_{|\mathcal{D}|}\}$, where $|\mathcal{D}|$ is the number of trajectories, and ξ is a trajectory defined as $(s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T)$ which is similar to the trajectory τ in RL, but removes the reward values at each time step. Note that the trajectory length T is a variable. In the following text, I use ξ_a and ξ_h to respectively denote trajectories by agent and human. Provided a reward function R, the score of a trajectory ξ, either from the expert or sampled by the agent, is evaluated as

$$c\,(\xi) = \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t). \tag{2.7}$$

This is similar to the definition of return in Eq. (2.1). However, note that in many maximum entropy IRL studies [58, 160, 161], an undiscounted form of return is used, i.e., the γ is set to 1. This thesis also adopts the maximum entropy IRL method (as will be explained later). Differently, it emphasizes the importance of using a discounted return in replicating human behavior.

Formulation of IRL for visual search. In the visual search process, the stimulus is a search image I_{search} and the target is specified using a target image I_{target}. The two images are pre-processed to have consistent sizes across the whole dataset. The eye movements are represented by a scanpath, i.e., a sequence of fixation points. Existing literature often sets the initial fixation at the center of the search image [49]. Following this design, I also consider that the human and the agent start from the center of the screen, which is not counted in the scanpath. The fixation point at step t is represented by its pixel coordinates on the search image, denoted by $f_t = (x_t, y_t)$. The duration of fixation is not considered in this study. The agent's observation at step t can be partial and history-dependent, and is determined by a certain observation function $o_t = O(I_{\text{search}}, I_{\text{target}}, (f_0, \ldots, f_{t-1}))$. Here the observation function $O(\cdot)$ is often designed to mimic human perceptions. An example is the belief map [55, 57] generated by pre-trained models. In this thesis, a cropped part of the search image is used as the observation. This keeps more information from the image and has a lower computational cost.

From the perspective of IRL, a scanpath is modeled as a trajectory. The initial state s_0 is the observation at the starting point o_0. The action a_t is the fixation f_t. The human scanpaths are treated as expert demonstrations. Note that human scanpaths are conditioned on the search image and the target image, so the dataset has an extended form: $\mathcal{D} = \{(\xi_1, I_{\text{search},1}, I_{\text{target},1}), \ldots\}$. The reward function is also conditioned on the images: $R(s, a | I_{\text{search}}, I_{\text{target}})$.

In many RL and IRL studies [54, 55, 59, 60], the action space is formulated by discretizing the space of the search image into a grid. This provides a smaller number of available actions. I denote the size of the search image as $\text{size}_{\text{search},x} \times \text{size}_{\text{search},y}$, and the size of the grid as $\text{size}_{\text{grid},x} \times \text{size}_{\text{grid},y}$. Then, the discretized representation $f_t^g = (x_t^g, y_t^g)$ of fixation point f_t is given by:

$$x_t^g = \lfloor x_t \text{size}_{\text{grid},x} / \text{size}_{\text{search},x} \rfloor, \qquad y_t^g = \lfloor y_t \text{size}_{\text{grid},y} / \text{size}_{\text{search},y} \rfloor. \tag{2.8}$$

Here $x_t^g \in \{0, \ldots, \text{size}_{\text{grid},x} - 1\}$ and $y_t^g \in \{0, \ldots, \text{size}_{\text{grid},y} - 1\}$. Inversely, given the discretized action f_t^g, a pixel-based representation can be approximately recovered, so that a scanpath from the agent can be compared to that from the human. This is done by using the coordinates at the center of the grids,

$$\hat{x_t} = (x_t^g + 0.5)\, \text{size}_{\text{search},x} / \text{size}_{\text{grid},x}, \qquad \hat{y_t} = (y_t^g + 0.5)\, \text{size}_{\text{search},y} / \text{size}_{\text{grid},y}. \tag{2.9}$$

One of the main targets of scanpath modeling is to make the agent's policy similar to human behavior. Given a testing dataset $\mathcal{D}_{\text{test}}$ and a similarity metric $m : \Xi \times \Xi \to \mathbb{R}$, one way to evaluate the similarity of agent's policy π is to sample one scanpath ξ_a for each $\xi_h \in \mathcal{D}_{\text{test}}$: $(\xi_h, I_{\text{search},h}, I_{\text{target},h}) \xrightarrow{\pi} \xi_a$, and then calculate the similarity value as:

$$M = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\xi_h, I_{\text{search}}, I_{\text{target}}) \in \mathcal{D}_{\text{test}}} m(\xi_a, \xi_h). \tag{2.10}$$

This considers the agent policy to be deterministic. For stochastic policies, a fixed number of agent scanpaths can be sampled to evaluate the expectation of the similarity value.

General IRL methods. Types of general IRL methods include maximum margin IRL [162], maximum entropy IRL [160], Bayesian IRL [163], and adversarial IRL [164]. Maximum entropy IRL is simple and allows imperfections in expert demonstrations, which suits the visual search task because of the noise and inconsistencies in human fixations. My proposed method adopts guided cost learning [161], one of its variants, and incorporates modifications.

In maximum entropy IRL, the demonstrations from the expert are assumed to be sampled from a distribution, where the probability for each demonstration is proportional to the exponential of their score, as formulated in Eq. (2.11) [160].

$$p(\xi_h) = \frac{1}{Z(\xi)} \exp\{c(\xi_h)\}, \tag{2.11}$$

where $Z(\xi) = \sum_{\xi \in \Xi} \exp\{c(\xi)\}$ is the partition function over all possible trajectories, denoted by Ξ. This probabilistic model allows noise and imperfections in human scanpaths. The training objective is to find a reward function R that maximizes the log-likelihood of observing the expert demonstrations. When the reward function is represented as a function approximator with parameters θ_r, the objective can be written as to maximize the following objective function:

$$\mathcal{L}(\theta_r) = \frac{1}{|\mathcal{D}|} \sum_{\xi_h \in \mathcal{D}} \log[p(\xi_h|\theta_r)]. \tag{2.12}$$

In Eq. (2.11), the partition function Z can be difficult to compute when the state and action space are large. Guided cost learning [161] provides a sample-based approach for calculating the objective function. I use $\mathcal{D}_a$ to denote a sampled set of agent trajectories, in which each trajectory ξ_a is sampled with probability $p(\xi_a)$. Note that this probability value $p(\xi_a)$ is dependent on the agent's policy. Then, the log-likelihood of a specific expert demonstration ξ_h can be derived as following:

$$\begin{aligned} \log[p(\xi_h|\theta_r)] =& c(\xi_h) - \log\{Z(\xi)\} \\ =& c(\xi_h) - \log\left\{ \frac{1}{|\mathcal{D}_a|} \sum_{\xi_a \in \mathcal{D}_a} \left[\frac{\exp(c(\xi_a))}{p(\xi_a)} \right] \right\}. \end{aligned} \tag{2.13}$$

In guided cost learning [161], the sample distribution $p(\xi_a)$ is learned to maximize an entropy-augmented objective, $\max \mathbb{E}_{p(\xi_a)}\left[c(\xi_a)\right] + \mathcal{H}\left[p(\xi_a)\right]$, where $\mathcal{H}\left[p(\xi_a)\right]$ is the differential entropy. When learning the reward model, the samples are augmented with the demonstrations and importance sampling is applied to all the trajectories. In this thesis, I show that some of the designs, such as demonstration augmentation, are not necessary for the visual search task.

IRL methods for visual search. Perhaps the first IRL application to scanpath modeling is reported in [55]. Generative adversarial imitation learning (GAIL) [164] is used in this study to learn from scanpaths from the COCO-Search18 dataset. For the observation model, the dynamic-contextual-belief is proposed to represent the state observation history in the search process. The contextual belief is built upon a Panoptic-FPN, a segmentation model pre-trained on the COCO2017 dataset. This observation model removes detailed textures and keeps a record of historical observations. A drawback is that the application is limited to searching in images similar to the COCO2017 dataset. Similarly, Chakraborty et al. [60] use GAIL for a webpage-based free-viewing task. A trained fixation density map is used to replace the dynamic-contextual-belief as state representation. Chen et al. [56] improve the method in [55] by an additional bias term in the reward function formulation and the employment of a Wasserstein generative adversarial network while using the original state representation. A recent study [57] focuses on the policy formulation and uses the option framework from the hierarchical RL to capture subtask switching behaviors in visual search.

The adversarial component in GAIL can make it difficult to train. It has been reported that GAIL is sensitive to its hyperparameters [165]. Some other visual search studies adopt IRL methods other than GAIL. In an early work [58], the maximum entropy IRL method is used to predict scanpaths collected from a classification task. A limitation is that the reward function is parameterized as a linear model based on pre-extracted features, which may not scale well to more complex scenes. Yang et al. [59] adopt the IQ-Learn algorithm [165], which learns a Q-value function instead of the reward function. A pre-trained model is used to process the state observation.

Despite the differences in policy formulation, base IRL method, and state representations, all the IRL studies mentioned above adopt a reward function that depends on state-action pairs. Specifically, the maximal entropy IRL study [58] explicitly learns a state-action-dependent reward function; the IQ-Learn study [59] learns a Q-value function to recover the reward function; the GAIL-based studies [55, 57, 60] train discriminators with state-action pairs, so even if the reward function is recoverable in special cases, it still depends on actions. This reliance on actions makes the reward predictions dynamic with respect to the search image and the target, and thus may not correctly capture the static motivation conditioned on the images. In contrast, this thesis proposes a decoupled reward function formulation that allows learning a static motivation representation. The learned representation is successfully applied to compare the effects of different parts of the stimuli.

Brain-inspired models for scanpath modeling.

The aforementioned supervised learning, RL, and IRL studies all use conventional ANN to predict human fixations. Despite the potential in helping understand brain mechanisms [29], brain-inspired models have seldom been applied to visual search modeling. There is a study that builds an attention prediction system where each component resembles the functions of a part of the brain [159]. However, the components are implemented using conventional ANNs, which do not resemble the biological neurons. As far as I know, a recent study [61] is the only one that applies SNN to scanpath prediction. Nevertheless, their SNN is obtained using conversion and surrogate gradient methods. In this thesis, my SNN is trained using R-STDP, thus being more biologically plausible and having better potential in relating brain structures and functions. This thesis provides the first application of an R-STDP-based SNN to visual search scanpath modeling.

2.3.2 Evaluation of Scanpath Predictions

A commonly used type of criterion for evaluating a scanpath prediction is its similarity to human scanpaths [51, 54, 55]. There are two challenges in measuring the similarity. One is the potential difference in scanpath lengths, which prevents the application of fixation-by-fixation distances. The other is the potentially redundant fixations. For example, consecutive fixations close to each other may need to be clustered into one fixation.

A simple way for evaluating the similarity is to convert scanpaths to attention maps and measure the difference between the two maps. Histogram intersection, correlation coefficient, information gain, and normalized scanpath saliency are some of the popular metrics [52, 59, 142, 166]. The problem is that the sequential information is discarded. Similarly, the Mannan linear distance [167] mentioned in [168] finds nearest neighbors to calculate the distance, which also neglects the order of fixations.

A basic method that preserves the sequential information is the Levenshtein distance [169]. The fixation space is discretized into grids, which are also known as areas of interest [168], each assigned a different letter. This transforms the two scanpaths to be compared into two strings, and then the minimum number of edits is calculated as the distance. ScanMatch [170] extends this metric to take into consideration the duration of fixations and relationships between areas, such as distance, color, and semantic pattern. Due to the lack of duration and semantic information, in this thesis, the Levenshtein similarity metric, instead of ScanMatch, is adopted as one of the metrics.

A general shortcoming of using discretized areas is that the representations of fixations near segmentations of areas could be far different from each other, while the fixations themselves are actually similar. In contrast, MultiMatch [168] is a metric that operates in the original pixel space of scanpaths. It considers scanpaths as sequences of two-dimensional vectors. A simplification step and an alignment of the scanpaths are performed to remove noise and extract meaningful fixations. MultiMatch reports five similarity values, respectively characterizing the average differences in shape (i.e., saccade vector difference), length, direction (i.e., angular difference), fixation position, and fixation duration. MultiMatch is also adopted in this thesis.

In target-present cases, the objective of finding the target introduces another type of metrics, different from similarity, that measure the success rate or speed of locating the target. An example is the area under curve where the curve is the estimated cumulative probability of fixating at the target object, as adopted in [55]. Gupta et al. [51] use a similar metric that counts the number of fixations in each scanpath to find the target image. Chen et al. [56] use the scanpath ratio that is based on the Euclidean distance between fixations and the target image. These metrics measure the performance of the agent in visual search. Since this thesis mainly focuses on simulating human behavior, this type of metrics is not adopted.

2.3.3 Applications of Visual Search

Visual search has a wide range of applications, including robot vision, autonomous driving, human-computer interface design, and mental healthcare [30, 48, 49]. In robot visual systems, object searching is an important task [171]. Imitating the saccade behavior of humans can help reproduce humans' efficiency in fixation selection [57, 59]. In human-computer interface design, visual search models can also reveal a person's intentions, the anticipation of which is crucial for human-computer interaction [49]. For mental healthcare, people with disorders and neurotypical people exhibit different patterns in search behavior. Visual search provides an economic and fast way for early diagnosis of disorders like ASD [150].

Through building visual search models, the attention of a person can be revealed, which is relevant to many daily tasks beyond visual search. In autonomous driving, the attention models of drivers can help develop assistance systems to improve driving safety [172]. In document design, the attention reveals whether the intended information is efficiently conveyed to the reader and helps improve the design [60].

This thesis applies visual search to the pattern analysis of ASD. A detailed review of ASD-related analysis based on eye movements is below. The new IRL method proposed in this thesis extracts static motivation representations for visual search, which is not constrained to mental healthcare. Future work may explore its application to other fields such as driver assistance.

2.3.4 ASD Analysis by Eye Movements

Diagnosis of autism spectrum disorder (ASD) can be costly [173]. Detection of ASD using more convenient observations is promising. A common understanding of ASD is atypical visual attention, with a reduced saliency towards social stimuli [174]. Many studies have investigated the eye movements of subjects with or without ASD to gain a better understanding of the behavior patterns of ASD and, further, to detect ASD based on eye movements. Here I review their representative examples, taking into consideration some studies where the human task is beyond the task of visual search, e.g., free-viewing [174], video watching.

I categorize these studies into three types: statistics-based behavior analysis, subject classification, and behavior prediction. Statistics-based studies examine pre-defined behavior patterns under different experimental conditions. An early work [175] uses the "Bubbles" method to compare the effects of different regions of stimuli on fixations in a face-viewing task. The comparison between an ASD group and the control group shows behavior patterns of people with ASD including an increased tendency to saccade away from fixated eye regions. In [30], a visual search dataset is collected from people with ASD, amygdala lesions, and healthy controls. The search image is specially constructed with social and non-social items to facilitate comparison of their contribution to eye movements. The statistics considered include simple ones like behavioral performance and fixation duration on social/non-social items, and also more sophisticated ones like percentage of social/non-social items visited before target detection, and the "target-relevant effect", which is defined as the difference between the percentage of target-congruent and target-incongruent items visited. One of the main findings is that the ASD group shows less target-relevant effects in early fixations in the scanpaths, indicating a slower orientation towards target-relevant items than the control group.

The subject classification studies mainly focus on distinguishing scanpaths from people with ASD and controls. An early work [176], although not for ASD classification, designs a number of features and trains an SVM to classify different groups of subjects. Similarly, Liu et al. [177] design a clustering-based saliency representation to be used by an SVM in the classification of people with ASD. Jiang and Zhao [173] also use SVM for subject classification, but the feature representation is extracted by a neural network trained to predict the difference between fixation maps. Eraslan et al. [150] use the scanpath trend to summarize the behaviors of each subject and use a Levenshtein distance-based clustering to classify subjects. In [138], a deep neural network model constructed with CNN and LSTM is built for ASD classification based on scanpath inputs. Beyond classification, these studies also provide an understanding of differences between groups of subjects, typically by altering the input features and looking at the classification performance. In [176], different subsets of features are used to train SVMs to examine their contribution to the classification. The most contributing features can be inferred to be affected by the disorders. In [177], a subset of the features is shown to offer a reasonably good classification performance. The corresponding image regions can thus be inferred to better reflect the differences in eye movements. In [173], important features for the SVM model are visualized to show an increased lower-level saliency and decreased social attention in the ASD group. In [138], it is discovered that the model performs better when there is a person in the stimulus image and worse when there is not or when the image shows a natural scene, which could indicate that the differences between ASD and control are mainly reflected in the scanpaths on person-present images.

The above two types of studies are helpful to better understand the behavior patterns of people with ASD and even to facilitate economic detection of ASD. However, there are limitations. The statistics-based methods are limited to relatively simple variables designed by researchers. More abstract properties of scanpaths, like the memory load, cannot be easily extracted and compared. Additionally, the detailed textures of the image stimuli can contribute to different eye movements but are often neglected (e.g., in [30], only the category of an item is kept). The subject classification studies, on the other hand, are capable of extracting complex features from the scanpaths. The shortcoming is that the classification target may not capture the mechanism for the generation of scanpaths.

In contrast, predictive models aim to reconstruct human behaviors given the same stimuli. A predictive model can help reveal how a specific group of subjects views the images, which provides a mechanism for group-wise comparison. As commented in [178], building predictive models can help with the diagnosis of ASD and also help design better content for people with ASD to view more easily. In [174], an SVM is built to predict fixation maps by weighting a set of pre-defined feature maps. Example results show that people with ASD exhibit lower object-level weights and higher pixel-level weights. Recent studies use deep neural networks to make the prediction. In [178], various networks, including SALICON which is based on VGG-16, are tried to predict the saliency maps. The models are pre-trained on normal datasets and fine-tuned on an ASD dataset. In [179], a CNN-based model is built for saliency prediction, with some of the hyperparameters designed with consideration of an ASD dataset. As far as I know, existing studies on predictive models for ASD-related analysis are concerned with saliency maps. So far, there are no scanpath-level predictive models that can capture not only the spatial attention but also the temporal decision policy of the subjects. This thesis is the first attempt to build a scanpath-level predictive model for ASD-related analysis.

Chapter 3

Spiking Variational Policy Gradient: A Novel Brain-Inspired Reinforcement Learning Algorithm

Among the main types of brain-inspired RL methods, R-STDP methods have the advantage in inference latency and biological plausibility, which facilitates analysis and understanding of brain mechanisms. However, most of the existing R-STDP methods use a shallow network structure with only one hidden layer [39, 45–47]. This deficiency in the capability of representing complex mappings hinders them from being applied to more challenging RL tasks. Extending R-STDP methods to more complex networks necessitates bridging the gap between the local learning rules and the RL task target. This chapter designs a new SNN structure and learning method to better bridge the gap and achieve better task performance. Specifically, this chapter proposes a recurrent and fully connected new network structure for SNN. Mean-field variational inference is adopted and a last-step approximation is proposed to derive the R-STDP learning rule for training the new network for policy gradient methods. This chapter first introduces the network design, and then presents the theoretical derivation of the policy inference and optimization methods.

This chapter includes work that has been published in a jointly-authored publication [74]. Among the materials included in this chapter, the author of this thesis completed the theoretical derivations and text. Shangqi Guo, Ying Fang, Zhaofei Yu, and Jian K. Liu contributed to the formulation of the ideas in the theoretical analysis. Shangqi Guo, Ying Fang, and Jian K. Liu contributed to the drawing of the sketch of the network, that is, Figure 2 in the publication, which corresponds to Figure 3.1 in this chapter.

3.1 Network Design

The network design involves the selection of the spiking neuron model, base component, and network structure. As reviewed in Chapter 2, SRM is a widely used model in the literature and is adopted in this thesis. As for the base component, previous studies have demonstrated the capability of WTA circuits in implementing the inference of a hidden Markov model [62] and hierarchical Bayesian inference [63], and policy distribution [47]. Therefore, I also adopt the WTA circuit as the base component of the network.

Existing R-STDP studies mainly use layered networks with only one hidden layer [39, 45–47]. This limitation hinders the networks from representing more complex mappings and being used to solve more challenging RL tasks. In this study, I design the network to be fully connected and recurrent. The recurrent design, as will be shown later, facilitates inference of a policy distribution. The full connection strengthens connections between neurons and enlarges the capacity. In this thesis, my network design is named as recurrent winner-take-all (recurrent WTA, RWTA) network.

The RWTA network is sketched in Figure 3.1. The RWTA network consists of some state neurons, some hidden WTA circuits, and one action WTA circuit. The firing probabilities of each state neuron encode one element of the state observation, while the firing states or probabilities of the action neurons can be used to generate the action decision. Recall that in a WTA circuit, only one neuron can fire at each spike time step. Here it is assumed that the number of actions in the RL task is finite, so each action in the task can be assigned to an action neuron, and the neuron that spikes is selected as the action decision. The adaptation to continuous action spaces or tasks with an infinite number of actions is left for future work. The network is fully connected, with all neurons from different circuits connected, but certain parts of the connections can be removed to create different network structures. The connections are symmetric, meaning the weight is shared by the two connected neurons. However, the connections that start from state neurons are unidirectional since the state neurons are not to be optimized.

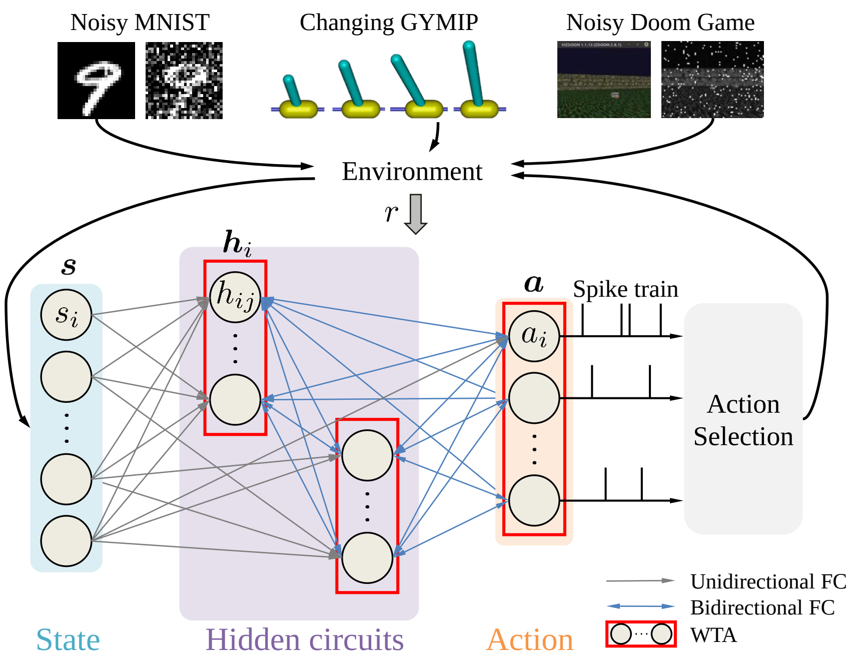


Figure 3.1: The illustration of the RWTA network structure. The RWTA network consists of the state input, a set of hidden WTA circuits, and an action WTA circuit as output. The environments (noisy MNIST, varied GYMIP, or noisy DOOM game) provide a reward r to modulate the fully-connected (FC) weights between neurons.

The state neurons are denoted as s_i $(i = 1, \dots, d_s)$; the action neurons are a_i $(i = 1, \dots, d_a)$; the j-th neuron in the i-th hidden circuit is h_{ij} $(i = 1, \dots, n_h,\ j = 1, \dots, d_h)$. Here d_s, d_h, and d_a are the sizes of the state observation and the WTA circuits, and n_h is the number of hidden WTA circuits. At each spike time step, each neuron has two properties: firing probability $q \in [0, 1]$ and binary firing status $v \in \{0, 1\}$. I use vectors to represent the values of groups of neurons, use h_i and h to denote the i-th hidden circuit and the entire set of hidden neurons, and use bold symbols with no subscript to denote all the neurons. For example, $\boldsymbol{q}_{h_i} := [q_{h_{i1}}, \dots, q_{h_{id_h}}]^{\mathrm{T}}$, $\boldsymbol{v}_h := [\boldsymbol{v}_{h_1}^{\mathrm{T}}, \dots, \boldsymbol{v}_{h_{n_h}}^{\mathrm{T}}]^{\mathrm{T}}$, and $\boldsymbol{q} = [\boldsymbol{q}_h^{\mathrm{T}}, \boldsymbol{q}_a^{\mathrm{T}}, \boldsymbol{q}_s^{\mathrm{T}}]^{\mathrm{T}}$. The total number of neurons is $N = n_h d_h + d_a + d_s$. The learnable parameters in the network are denoted as $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ for the synapse weights and $\boldsymbol{b} \in \mathbb{R}^N$ for the self-activation parameters; the columns and rows of $\boldsymbol{W}$ are arranged by h, a, s and $\boldsymbol{b}$ is arranged according to h, a, s. Note that the state neurons have zero intrinsic excitabilities, i.e., $\boldsymbol{b}_s = \boldsymbol{0}$. I use θ to refer to the parameters of the policy, i.e., $\theta = \langle \boldsymbol{W}, \boldsymbol{b} \rangle$.

When applied to an RL task, the RWTA network is simulated for a fixed number N_{iter} of spike time steps (e.g., 100) for each RL time step t. Before the simulation for each RL step, the state of the whole network is initialized by setting the firing probabilities of state neurons to an encoding of the RL state and setting the other neurons according to a $[0, 1]$ random uniform distribution. After the simulation, the final firing state of the action WTA circuit corresponds to the action decision. The RL reward r_t can be used to update the network parameter.

As far as I know, the most similar structure in previous studies is the layered WTA network proposed in [47], which considers one hidden layer of WTA circuits and the connections between layers are recurrent. Compared to this structure, my design features extra connections between types of neurons. Specifically, there are connections between hidden WTA circuits and between state and action neurons. As will be shown later in section 4.3.4, these connections contribute to the improvement in task performance.

3.2 Policy Inference

3.2.1 The Definition of Policy Function

I set the RL policy to be a probability distribution over the action space, which is assumed to consist of a finite number of actions, and define it with an energy function $E(\boldsymbol{v})$. The energy function-based design of the policy function has been adopted by existing studies [47, 180]. Specifically,

$$\pi(\boldsymbol{v}_a|s) = \sum_{\boldsymbol{v}_h} p(\boldsymbol{v}_a, \boldsymbol{v}_h|s), \tag{3.1}$$

$$p(\boldsymbol{v}_a, \boldsymbol{v}_h|s) := \frac{1}{Z(s)} \exp\{E(\boldsymbol{v})\}, \tag{3.2}$$

$$E(\boldsymbol{v}) := \boldsymbol{v}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{v} + \boldsymbol{b}^{\mathrm{T}} \boldsymbol{v}, \tag{3.3}$$

where $Z(s) = \sum_{\boldsymbol{v}'_h, \boldsymbol{v}'_a} \exp\{E(\boldsymbol{v}')\}$ is the normalization. As shown in the above equations, the policy distribution is calculated as the marginal distribution of $\boldsymbol{v}_a$. Although the energy function is linear, the normalization operation makes the energy-based policy function capable of representing complex distributions [180]. This formulation makes the policy function similar to the SRM model, i.e., Eqs (2.2) and (2.3). More importantly, as will be shown later, this formulation of policy is equivalent to the fixed point of the RWTA network.

The policy representation Eq. (3.1) is computable in principle. However, when the number of hidden neurons is large, it can be intractable in practice. To address this problem, I adopt mean-field inference to derive an approximation $\hat{p}(\boldsymbol{v}_a, \boldsymbol{v}_h|s)$ of the probability function of action-hidden states $p(\boldsymbol{v}_a, \boldsymbol{v}_h|s)$ (see section 3.2.3 below).

3.2.2 Validity of Policy Approximation

The above approximation can induce a change to the expected return. Before diving into the details of the approximation, I first analyze the relationship between the approximation and the change. To do this, I equivalently transform the original objective $J(\pi)$ into $\log J(\pi)$. I use τ_+ to denote the trajectories in which hidden states $\boldsymbol{v}_h$ are incorporated with actions. I use the subscript "est" to refer to the probabilities or distributions of trajectories under the approximated policy function. Then, the following lower bound of the objective when the policy approximation is applied can be obtained:

$$
\begin{aligned}
\log J(\pi) &= \log \left[\sum_{\tau_+} p_\pi(\tau_+) \sum_{t=0}^{T-1} \gamma^t r_t \right] \\
&\geq \mathbb{E}_{\text{est}} \left[\log \sum_{t=0}^{T-1} \gamma^t r_t \right] - D_{\text{KL}} \left[p_{\text{est}}(\tau_+) \parallel p_\pi(\tau_+) \right],
\end{aligned}
\tag{3.4}
$$

where D_{KL} is the Kullback-Leibler (KL) divergence. For deterministic environments, target $\mathbb{E}_{\text{est}}[\log \sum_{t=0}^{T-1} \gamma^t r_t]$ is equivalent to $\mathbb{E}_{\text{est}}[\sum_{t=0}^{T-1} \gamma^t r_t]$, which is the original expected return. In stochastic environments, the equivalence depends on the transition function P. When the equivalence holds, the above Eq. (3.4) indicates that, by minimizing the KL divergence between the approximated function $\hat{p}(\boldsymbol{v}_a, \boldsymbol{v}_h|s)$ and the original function $p(\boldsymbol{v}_a, \boldsymbol{v}_h|s)$, the lower bound of the original expected return can be maximized.

3.2.3 Policy Mean-Field Inference

I adopt variational inference, which has been used in existing studies [47, 64], to derive an approximation of the policy function. I use a variational distribution $\hat{p}(\boldsymbol{v}_a, \boldsymbol{v}_h|s)$ to approximate $p(\boldsymbol{v}_a, \boldsymbol{v}_h|s)$, and assume that the firing states of all circuits are independent to each other. This leads to a decomposition of $\hat{p}$, i.e., $\hat{p}(\boldsymbol{v}_a, \boldsymbol{v}_h|s) := \hat{p}(\boldsymbol{v}_a|s)\hat{p}(\boldsymbol{v}_{h_1}|s) \cdots \hat{p}(\boldsymbol{v}_{h_{n_h}}|s)$, where $\hat{p}(\boldsymbol{v}_{h_1}|s) := \boldsymbol{q}_{h_1}^{\mathrm{T}} \boldsymbol{v}_{h_1}$, $\ldots$, $\hat{p}(\boldsymbol{v}_a|s) = \boldsymbol{q}_a^{\mathrm{T}} \boldsymbol{v}_a$.

By minimizing the KL divergence between $\hat{p}$ and p, i.e., $D_{\text{KL}}(s) \doteq D_{\text{KL}}[\hat{p}(\boldsymbol{v}_a, \boldsymbol{v}_h|s) \| p(\boldsymbol{v}_a, \boldsymbol{v}_h|s)]$, I get the following mean-field inference equation [181] for each hidden or action neuron i:

$$
q_i = \frac{1}{Z(q_{G(i)})} \exp \left\{ \boldsymbol{w}_{\text{row},i}^{\mathrm{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},i}^{\mathrm{T}} \boldsymbol{q} + b_i \right\}, \tag{3.5}
$$

where $i = 1, \ldots, (n_h d_h + d_a)$, $G(i)$ is the set of indices of the neurons in the same circuit as neuron i, $Z(q_{G(i)}) = \sum_{j \in G(i)} \exp\{\boldsymbol{w}_{\text{row},j}^{\text{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},j}^{\text{T}} \boldsymbol{q} + b_j\}$, and $\boldsymbol{w}_{\text{row},i}$ and $\boldsymbol{w}_{\text{col},i}$ are respectively the i-th row and column of matrix $\boldsymbol{W}$ (in the shape of a column vector), which corresponds to the synapses connected to neuron i. b_i is the i-th element in vector $\boldsymbol{b}$.

To get the policy distribution $\pi(\boldsymbol{v}_a|s)$, which is approximated by $\boldsymbol{q}_a$, I can solve Eq. (3.5) to get $\boldsymbol{q}$ and then extract its elements corresponding to $\boldsymbol{q}_a$. Eq. (3.5) can be seen as an iteration process by regarding the $\boldsymbol{q}$ on the right side as a constant vector. In practice, one numerical method to get the solution $\boldsymbol{q}$ is to initialize $\boldsymbol{q}$ with random numbers and then repeat updating it with Eq. (3.5) until numerical convergence. Although there is no theoretical guarantee of convergence, I demonstrate in experiments that it converges in most cases (see section 4.3.1).

3.2.4 Policy Inference with RWTA Network

Now I show that the fixed point of the RWTA network equals the approximated policy inference above. That is, the iterative method above for policy inference can be implemented with the RWTA network.

I assume that the internal inhibitory neuron in the WTA circuits makes the overall firing rates of the network (excluding the state neurons) a constant value $\hat{\rho} \in (0, 1)$. With this assumption, I let the firing probabilities encode $\rho_i(l) = \hat{\rho} q_i$. Then the policy inference function Eq. (3.5) is transformed to

$$\rho_i = \hat{\rho} \exp\left\{ \boldsymbol{w}_{\text{row},i}^{\text{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},i}^{\text{T}} \boldsymbol{q} + b_i - \log \sum_{j \in G(i)} \exp\{\boldsymbol{w}_{\text{row},j}^{\text{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},j}^{\text{T}} \boldsymbol{q} + b_j\} \right\}. \tag{3.6}$$

Then, consider the neuron model Eqs. (2.2)and (2.3) in the RWTA network, I assign w_j with the synaptic weights $w_{ij} + w_{ji}$, and design $\kappa(y)$ such that $\int_0^\infty \kappa(y) \mathrm{d}y = 1/\hat{\rho}$. This transforms the probability values $\boldsymbol{q}$ in Eq. (3.6) into membrane potential $u(l)$, leading to the following spike-based inference function

$$\begin{aligned} \rho_i(l) &= \hat{\rho} \exp\left\{ u_i(l) - \log \sum_{j \in G(i)} \exp(u_j(l)) \right\}, \\ u_i(l) &= \sum_{j \in N(i)} w_{ij} \int_0^\infty \kappa(y) S_{ij}(l - y) \mathrm{d}y + b_i. \end{aligned} \tag{3.7}$$

Eq. (3.7) shows the way the RWTA network iterates its membrane potentials and firing probabilities. When a fixed point is reached, the firing probabilities give the solution to the policy inference function Eq. (3.5). This shows that the RWTA network designed above can perform the approximated policy inference. Note that it is biologically plausible as it conforms to the definition of the SRM neuron model.

3.3 Policy Optimization

The policy optimization concerns the update of network parameters θ and relies on a base RL algorithm. Here I select REINFORCE as the base algorithm, because it is the base of many popular algorithms like A3C [25] and PPO [65], and it has a simple policy gradient formulation that can simplify the derivation of the method. I derive the learning method and build its relationship to the R-STDP framework. Then I extend the method to other base RL algorithms like PPO.

3.3.1 Policy Optimization for REINFORCE

In REINFORCE, the policy gradient is calculated according to the following equation [1]:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\gamma^t r_t \sum_{k=0}^{T-1}\nabla_\theta \log \pi_\theta(a_k|s_k)\right], \tag{3.8}$$

which contains $\nabla_\theta \log \pi_\theta(a_k|s_k)$, differential of logarithm of the policy function. Based on the policy approximation $\hat{p}(\boldsymbol{v}_a, \boldsymbol{v}_h|s)$ in previous subsection, this differential stands for $\nabla_\theta \log(\boldsymbol{q}_{ha})$. According to the policy inference function Eq. (3.5), this differential can be calculated as shown in the following theorem.

Theorem 3.1. *(Precise optimization rule) The* ***precise differential*** *of $\boldsymbol{q}_{ha}$ to a certain synapse weight w_{jk} and the self-activation parameter b_j is*

$$\begin{aligned}\frac{\partial \boldsymbol{q}_{ha}}{\partial w_{jk}} &= \boldsymbol{M}\big(\boldsymbol{U}_{jk} + \boldsymbol{U}_{kj}\big)\boldsymbol{q} + \boldsymbol{M}(\boldsymbol{W} + \boldsymbol{W}^{\mathrm{T}})\frac{\partial \boldsymbol{q}}{\partial w_{jk}},\\ \frac{\partial \boldsymbol{q}_{ha}}{\partial b_j} &= \boldsymbol{M}\boldsymbol{b} + \boldsymbol{M}(\boldsymbol{W} + \boldsymbol{W}^{\mathrm{T}})\frac{\partial \boldsymbol{q}}{\partial b_j},\end{aligned} \tag{3.9}$$

where $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{q}_{ha})[-\boldsymbol{G}_{ha}\mathrm{diag}(\boldsymbol{q}) + \boldsymbol{D}_{\mathrm{sel}}]$, $\boldsymbol{G}_{ha}$ is a logical matrix with shape $(n_h d_h + d_a) \times N$ where 1 elements indicate the two neurons (column index and row index) are in the same WTA circuit, $\boldsymbol{D}_{\mathrm{sel}}$ is a logical matrix that selects the first $(n_h d_h + d_a)$ elements in a vector with length

N, i.e., $\boldsymbol{D}_{\text{sel}} = \begin{bmatrix} \boldsymbol{I}_{(n_h d_h + d_a)} & \boldsymbol{O}_{(n_h d_h + d_a) \times d_s} \end{bmatrix}$, and $\boldsymbol{U}_{jk}$ is a logical matrix with shape $N \times N$ where only the jk-th element is 1.

Proof. The mean-field policy inference function is

$$q_i = \frac{1}{Z(q_{G(i)})} \exp\{\boldsymbol{w}_{\text{row},i}^{\text{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},i}^{\text{T}} \boldsymbol{q} + b_i\}, \tag{3.10}$$

where $Z(q_{G(i)}) = \sum_{j \in G(i)} \exp\{\boldsymbol{w}_{\text{row},i}^{\text{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},i}^{\text{T}} \boldsymbol{q} + b_i\}$, $i = 1, \ldots, (n_h d_h + d_a)$, $G(i)$ is the set of indices of the neurons in the same circuit as neuron i, and $\boldsymbol{w}_{\text{row},i}$ and $\boldsymbol{w}_{\text{col},i}$ are respectively the i-th row and column of matrix $\boldsymbol{W}$ (in the shape of a column vector), which corresponds to the synapses connected to neuron i. b_i is the i-th element in vector $\boldsymbol{b}$.

For each w_{jk}, There is

$$\begin{aligned} \frac{\partial q_i}{\partial w_{jk}} =& - Z^{-2}(q_{G(i)}) \frac{\partial Z(q_{G(i)})}{\partial w_{jk}} \exp\left\{\boldsymbol{w}_{\text{row},i}^{\text{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},i}^{\text{T}} \boldsymbol{q} + b_i\right\} \\ &+ Z^{-1}(q_{G(i)}) \exp\left\{\boldsymbol{w}_{\text{row},i}^{\text{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},i}^{\text{T}} \boldsymbol{q} + b_i\right\} \\ &\cdot \sum_{m=1}^{N} \left[\frac{\partial (w_{im} + w_{mi})}{\partial w_{jk}} q_m + (w_{im} + w_{mi}) \frac{\partial q_m}{\partial w_{jk}}\right] \\ =& - q_i Z^{-1}(q_{G(i)}) \frac{\partial Z(q_{G(i)})}{\partial w_{jk}} + q_i \sum_{m=1}^{N} \left[\frac{\partial (w_{im} + w_{mi})}{\partial w_{jk}} q_m + (w_{im} + w_{mi}) \frac{\partial q_m}{\partial w_{jk}}\right]. \end{aligned} \tag{3.11}$$

For the term $\frac{\partial Z(q_{G(i)})}{\partial w_{jk}}$, there is

$$\begin{aligned} \frac{\partial Z(q_{G(i)})}{\partial w_{jk}} =& \frac{\partial}{\partial w_{jk}} \left\{ \sum_{m \in G(i)} \exp\left\{\boldsymbol{w}_{\text{row},m}^{\text{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},m}^{\text{T}} \boldsymbol{q} + b_m\right\} \right\} \\ =& \sum_{m \in G(i)} \left\{ \exp\left[\boldsymbol{w}_{\text{row},m}^{\text{T}} \boldsymbol{q} + \boldsymbol{w}_{\text{col},m}^{\text{T}} \boldsymbol{q} + b_m\right] \right. \\ & \left. \cdot \sum_{n=1}^{N} \left[\frac{\partial (w_{mn} + w_{nm})}{\partial w_{jk}} q_n + (w_{mn} + w_{nm}) \frac{\partial q_n}{\partial w_{jk}}\right] \right\}. \end{aligned} \tag{3.12}$$

So there is

$$\begin{aligned} \frac{\partial q_i}{\partial w_{jk}} =& - q_i \sum_{m \in G(i)} \left\{ q_m \sum_{n=1}^{N} \left[\frac{\partial (w_{mn} + w_{nm})}{\partial w_{jk}} q_n + (w_{mn} + w_{nm}) \frac{\partial q_n}{\partial w_{jk}}\right] \right\} \\ &+ q_i \sum_{n=1}^{N} \left[\frac{\partial (w_{in} + w_{ni})}{\partial w_{jk}} q_n + (w_{in} + w_{ni}) \frac{\partial q_n}{\partial w_{jk}}\right]. \end{aligned} \tag{3.13}$$

Similarly, for each b_j, there is

$$\frac{\partial q_i}{\partial b_j} = -q_i \sum_{m \in G(i)} \left\{ q_m \left[\frac{\partial b_m}{\partial b_j} + \sum_{n=1}^{N} (w_{mn} + w_{nm}) \frac{\partial q_n}{\partial_{b_j}} \right] \right\} + q_i \left[\frac{\partial b_i}{\partial b_j} + \sum_{n=1}^{N} (w_{in} + w_{ni}) \frac{\partial q_n}{\partial_{b_j}} \right]. \tag{3.14}$$

By respectively arranging Eq. (3.13) and Eq. (3.14) for each q_i into vectors, and combining the terms into matrices, Eq. (3.9) can be obtained. □

Theorem 3.1 reveals that the required differential can be obtained by solving the matrix equations Eq. (3.9). However, this involves the calculation of the pseudo-inverse of $\boldsymbol{M}(\boldsymbol{W} + \boldsymbol{W}^{\mathrm{T}})$, the shape of which is $(n_h d_h + d_a) \times N$. Therefore, the computational complexity is over $O((n_h d_h + d_a)^3)$, which can be intractable in practice when the number of hidden and action neurons is large.

Therefore, I propose to obtain an approximated solution of Eq. (3.5). As will be shown later, this approximation can still get satisfying results in the experiments. It can also be implemented in the R-STDP framework, so it has the advantage of being biologically plausible. My idea for the approximation is to regard the $\boldsymbol{q}$ on the right side of the policy inference function Eq. (3.5) as a constant on the network parameters. By doing so, the differential only concerns the last step in the inference process, where the status of each neuron is only affected by its neighboring neurons. Thus, the differential on a certain connection or neuron only depends on information from the connected neurons, which makes possible the link between the local rules and the global objective. The result is presented in the following theorem.

Theorem 3.2. *(Approximate optimization rule) The* ***approximate differentials*** *of firing rate q_i with respect to $\boldsymbol{W}$ and $\boldsymbol{b}$ are:*

$$\begin{aligned} \frac{\partial \log(q_i)}{\partial \boldsymbol{W}} =& (\boldsymbol{U}_{i:}\mathrm{diag}(\boldsymbol{q}) + \mathrm{diag}(\boldsymbol{q})\boldsymbol{U}_{:i}) - \mathrm{diag}(\boldsymbol{q})(\boldsymbol{U}_{G(i):} + \boldsymbol{U}_{:G(i)})\mathrm{diag}(\boldsymbol{q}), \\ \frac{\partial \log(q_i)}{\partial \boldsymbol{b}} =& \boldsymbol{u}_i - \mathrm{diag}(\boldsymbol{q})\boldsymbol{u}_{G(i)}, \end{aligned} \tag{3.15}$$

where $i \in \{1, \ldots, (n_h d_h + d_a)\}$, $\boldsymbol{U}$ is a $N \times N$ logical matrix and $\boldsymbol{u}$ is a length-N logical vector, whose subscripts indicate the positions of elements with value 1. *$G(i)$ is the set of indices of neurons in the same circuit as neuron i. Symbol ":" means the entire row/column.*

Proof. The condition is the same as that in the proof of Theorem 1. The approximate differentiation of firing rate q_i with respect to w_{jk} and b_j are:

$$\begin{aligned}\frac{\partial \log(q_i)}{\partial w_{jk}} =& \sum_{m=1}^{N}\left[\frac{\partial(w_{im}+w_{mi})}{\partial w_{jk}}q_m\right] \\ & - \frac{1}{Z(q_{G(i)})}\sum_{m\in G(i)}\left[\exp\{\boldsymbol{w}_{\text{row},m}^{\mathrm{T}}\boldsymbol{q}+\boldsymbol{w}_{\text{col},m}^{\mathrm{T}}\boldsymbol{q}+b_m\}\cdot\sum_{n=1}^{N}\frac{\partial(w_{mn}+w_{nm})}{\partial w_{jk}}q_n\right] \\ =& \sum_{m=1}^{N}\left[\frac{\partial(w_{im}+w_{mi})}{\partial w_{jk}}q_m\right] - \sum_{m\in G(i)}\left[q_m\cdot\sum_{n=1}^{N}\frac{\partial(w_{mn}+w_{nm})}{\partial w_{jk}}q_n\right],\end{aligned} \tag{3.16}$$

$$\begin{aligned}\frac{\partial \log(q_i)}{\partial b_j} =& \frac{\partial b_i}{\partial b_j} - \frac{1}{Z(q_{G(i)})}\cdot\sum_{m\in G(i)}\left[\frac{\partial b_m}{\partial b_j}\cdot\exp\{\boldsymbol{w}_{\text{row},m}^{\mathrm{T}}\boldsymbol{q}+\boldsymbol{w}_{\text{col},m}^{\mathrm{T}}\boldsymbol{q}+b_m\}\right] \\ =& \frac{\partial b_i}{\partial b_j} - \sum_{m\in G(i)}\left[q_m\frac{\partial b_m}{\partial b_j}\right].\end{aligned} \tag{3.17}$$

Similar to the proof of Theorem 1, by respectively arranging Eq. (3.16) and Eq. (3.17) for each q_i into vectors, and combining the terms on the right side into matrices, Eq. (3.15) can be obtained. □

According to Theorem 3.2, at the last step of each simulation for a certain RL time step, given the firing state $\boldsymbol{v}$ of the RWTA network, the corresponding REINFORCE policy gradient can be obtained:

$$\nabla J(\pi) = \sum_t \gamma^t r_t\left[\sum_{i=1}^{n_h}\boldsymbol{v}_{h_i}^{\mathrm{T}}\nabla(\log\boldsymbol{q}_{h_i})+\boldsymbol{v}_a^{\mathrm{T}}\nabla(\log\boldsymbol{q}_a)\right], \tag{3.18}$$

where $\nabla\log\boldsymbol{q}_{h_i}$ and $\nabla\log\boldsymbol{q}_a$ are respectively the vectors of $\nabla\log q_{h_{ij}}$ and $\nabla\log q_{a_i}$.

3.3.2 Policy Optimization with R-STDP

Now I show how this policy gradient can be implemented with R-STDP. Specifically, this means to design a set of $\langle W_{\text{pre}}, W_{\text{post}}, A_+(w_{ij}), A_-(w_{ij})\rangle$ in the R-STDP framework. I make the following settings to the R-STDP for two arbitrarily connected neurons i and j.

$$\langle W_{\text{pre}}, W_{\text{post}}, A_+(w_{ij}), A_-(w_{ij})\rangle = \left\langle v_i, v_j, -\frac{1}{\hat{\rho}}, -\frac{1}{\hat{\rho}}\right\rangle, \tag{3.19}$$

The expectation of the frequency of spikes in a spike train S_i equals the firing probability ρ_i. That is, $\mathbb{E}[S_i(l)] = \rho_i$ and $\mathbb{E}[\int_0^\infty A_+(w_{ij})S_i(l-y)\mathrm{d}y] = A_+(w_{ij})\rho_i$. Then, there is the following transformed formulation of my R-STDP rule:

$$\mathbb{E}[R(l)\text{STDP}(l)] = R' \left[\rho_j \left(v_i - \frac{\rho_i}{\hat{\rho}} \right) + \rho_i \left(v_j - \frac{\rho_j}{\hat{\rho}} \right) \right], \tag{3.20}$$

where R' is a signal about the environment reward for the considered simulation period. Note that, for the self-excitation parameter b, it can be regarded as the weight of a connection from an always-firing neuron and that the post-synaptic part of the STDP rule is omitted. The corresponding learning rule is $\Delta b_i = R'[v_i - \rho_i/\hat{\rho}]$.

Then, for the policy gradient Eq. (3.15, 3.18) that is derived from the global objective, they can be reorganized according to the network parameters as follows:

$$\begin{aligned} \frac{\partial J(\pi)}{\partial w_{ij}} =& \sum_t \gamma^t r_t [q_i(v_j - q_j) + q_j(v_i - q_i)], \\ \frac{\partial J(\pi)}{\partial b_i} =& \sum_t \gamma^t r_t (v_i - q_i). \end{aligned} \tag{3.21}$$

As shown, the two equations, Eq. (3.20) and Eq. (3.21), are equivalent. By using Monte-Carlo sampling methods, R' can be made equal to $\sum_t \gamma^t r_t$. By scaling the optimization step size with $\hat{\rho}$, the difference in the overall firing rate $\hat{\rho}$ can be removed. This means that the R-STDP rule defined in Eq. (3.19) can represent the approximated policy gradient on the RWTA network.

3.4 Algorithm

So far, I derive a variational policy gradient method where inference and optimization are implemented with the spiking RWTA network and an R-STDP rule. I name it spiking variational policy gradient (SVPG). Note that this method is based on the REINFORCE algorithm. The full process of SVPG is summarized in Algorithm 1.

In SVPG, the number of state neurons d_s and the number of action neurons d_a depend on the task setting. For example, in MNIST classification, the state observation is an image with a size of 28×28, so d_s is 784. The number of actions d_a is 10, corresponding to the number of available classes. The discount factor γ determines the extent to which future rewards are important, and is specified by the RL task. The hyperparameters of SVPG mainly include the inference iteration number N_{iter}, the learning rate η, and the network shape n_h and d_h. A larger iteration number N_{iter} means to simulate the SNN for more steps to obtain an RL action

Algorithm 1 SVPG with REINFORCE

Parameter: Discount factor γ. Training episode number N_{epi}. Inference iteration number N_{iter}. Learning rate η. Network shape n_h, d_h, d_a, d_s.

Output: RWTA Network parameter θ.

```
1: Initialize $\theta$ to zero.
2: for Episode = 1, ..., $N_{\text{epi}}$ do
3:   Clear memory buffer $\mathcal{D}$.
4:   for Training step $t = 1, \ldots, T$ do
5:     Observe and encode state $s_t$.
6:     Randomly initialize $\boldsymbol{q}_a$ and $\boldsymbol{q}_h$.
7:     Iterate Eq.(3.7) for $N_{\text{iter}}$ spike time steps. {Inference}
8:     Use $\boldsymbol{v}_a$ to generate $a_t$.
9:     Perform $a_t$, observe reward $r_t$ and new state $s_{t+1}$.
10:    Store $\langle s_t, a_t, r_t, s_{t+1}, \boldsymbol{q}, \boldsymbol{v} \rangle$ into $\mathcal{D}$.
11:  end for
12:  Get data from $\mathcal{D}$.
13:  Calculate gradient using Eq. (3.21). {Optimization}
14:  Update $\theta \leftarrow \theta + \eta \nabla \theta$ .
15: end for
```

decision, which can stabilize the policy distribution, but also increase the computational cost. A larger learning rate changes the learnable parameters with larger steps, which can increase training speed but meanwhile make the training process unstable. The number of hidden WTA circuits n_h and the size of the hidden WTA circuits d_h determine the capacity of the network.

In Algorithm 1, the RWTA network is first initialized with zero weights before training. Note that this does not make the hidden and action neurons silent because the WTA circuits ensure an overall firing probability of the neurons in a circuit. The algorithm runs in episodes. For each episode, the memory buffer $\mathcal{D}$ collects the transition data at each step, which is used for updating the RWTA network (step 13). For each step t, the RWTA network is simulated for N_{iter} steps to produce the action decision. Since the learning rule of SVPG requires the firing state and firing probability of neurons, $\boldsymbol{q}$ and $\boldsymbol{v}$ need to be stored with the transition data in the memory buffer $\mathcal{D}$.

3.5 Practical Considerations

There may be two problems with SVPG in practical application. (1) The simulation of spike trains in the RWTA network can be computationally expensive, particularly for general devices, such as graphics processing unit (GPU). (2) SVPG is derived for the REINFORCE algorithm, which is not efficient and is not popularly used in recent RL studies. Here I provide solutions to these two potential problems.

3.5.1 Rate-Based Approximation

For the computational cost problem, I propose a rate-based approximation of SVPG. In the approximation, the evolution of neurons' firing probabilities is directly calculated by the policy inference function Eq. (3.5), i.e., without the intermediate simulation of the spike trains. In this way, the computational cost can be reduced. However, this approximation also removes the random noise in firing probabilities caused by spike trains, which could be important to the overall performance. For this deficiency, I add Gaussian noise (with standard deviation $\sigma = 0.02$) to the firing probability values in each iteration of the firing probabilities. As will be shown later (section 4.3.1), this rate-based approximation can significantly reduce the computational cost while producing similar training and perturbation results to the original implementation.

3.5.2 Extension to Other Base RL Algorithms

The RL field has seen many advances in algorithms that bring improvements to training efficiency, scalability, etc. Extending SVPG to these RL algorithms can facilitate the test or application to more challenging scenarios. Here I propose methods to achieve this extension. I consider the PPO-clip algorithm [65], which has been widely used in RL studies and is generally considered faster and better at solving complex tasks like DOOM than REINFORCE. I also consider the extension to value-based RL algorithms, which are another major branch besides policy gradient.

For value-based algorithms like DQN, the network is required to output a number of state-action values [7]. The firing rates of the action neurons can be used to approximate their firing probabilities q_{a_i}, and then transformed to the state-action value with a mapping like $Q_{a_i} = \tan\{q_{a_i}\pi - \pi/2\}$. Suppose a loss function $\mathrm{Loss}(Q_{a,i})$ on the state-action value is defined in the base RL algorithm, its differential can be decomposed into two parts according to the chain rule $\partial \mathrm{Loss}(Q_{a_i})/\partial\theta = \frac{\partial \log(q_{a_i})}{\partial\theta} \cdot \frac{\partial \mathrm{Loss}(Q_{a_i})}{\partial \log(q_{a_i})}$. The first part has been derived in Eq. (3.15), and the second part can be calculated in practice using deep learning libraries such as PyTorch [182].

Algorithm 2 SVPG for PPO-clip

Parameter: Discount factor γ. Training episode number N_{epi}. Inference iteration number N_{iter}. Learning rate η. PPO epoch number N_{PPO}. Network shape n_h, d_h, d_a, d_s.
Output: RWTA Network parameter θ.

```
1: Initialize $\theta$ to zero. Initialize the critic network.
2: for Episode = 1, ..., $N_{\text{epi}}$ do
3:   Clear memory buffer $\mathcal{D}$.
4:   for Training step $t$ = 1, ..., $T$ do
5:     Observe and encode state $s_t$.
6:     Randomly initialize $\boldsymbol{q}_a$ and $\boldsymbol{q}_h$ and normalize them at circuit-level.
7:     Iterate Eq. (3.7) for $N_{\text{iter}}$ spike time steps. {Inference}
8:     Use $\boldsymbol{v}_a$ to generate $a_t$. Perform $a_t$, observe reward $r_t$ and new state $s_{t+1}$.
9:     Store $\langle s_t, a_t, r_t, s_{t+1}, \boldsymbol{q}, \boldsymbol{v} \rangle$ into $\mathcal{D}$.
10:  end for
11:  Get data from $\mathcal{D}$. Backtrack reward $R = \sum_t \gamma^t r_t$.
12:  Update critic network. Use the critic to generate an advantage value $A$.
13:  Store checkpoint $\theta_{\text{old}}$, $\boldsymbol{v}_{\text{old}}$ .
14:  for PPO epoch num = 1, ..., $N_{\text{PPO}}$ do
15:    Update $\theta$ using $A$, $\mathcal{D}$, $\theta_{\text{old}}$, $\boldsymbol{v}_{\text{old}}$ and Eq. (3.21), Eq. (3.22). {Optimization}
16:  end for
17: end for
```

For the PPO-clip algorithm, the learning target is [65]

$$\begin{aligned} J(\pi_\theta) &= \mathbb{E}_t \left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right], \\ r_t(\theta) &= \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, \end{aligned} \tag{3.22}$$

where π_θ is the current policy, $\pi_{\theta_{\text{old}}}$ is an old policy with checkpoint parameter θ_{old}, and ϵ is a hyperparameter. Similar to the value function representation, the differential of the policy distribution can be transformed into the one I derived earlier: $\frac{\partial q_i}{\partial \theta} = \frac{\partial \log(q_i)}{\partial \theta} \cdot q_i$. Note that there is a difference between conventional ANNs and the RWTA network when dealing with θ_{old}. The differential for the RWTA network, i.e., Eq. (3.21) requires the firing states of the neurons, which can be different in different simulations. Thus the checkpoint θ_{old} needs to include both the network parameters and the firing states $\boldsymbol{v}$. When updating the network with Eq. (3.21), the v values are from the checkpoint, and the q values are from the current policy instantiation. This extension reduces the biological plausibility of the SVPG method since it uses information from a previous state of the network; however, this is inevitable for most base RL algorithms that use the target network technique [7]. A sketch of the SVPG algorithm for PPO-clip is given in Algorithm 2.

3.6 Chapter Summary

In this chapter, I designed the RWTA network and showed that its fixed point corresponds to an approximated solution of the policy inference function for an energy-based policy formulation. I derived SVPG, an approximated optimization rule for the REINFORCE base algorithm, and showed its equivalence to the R-STDP framework. I further proposed extensions of SVPG to other base RL algorithms, including the PPO algorithm.

Chapter 4

Evaluation of SVPG on Benchmark RL Tasks

The previous chapter designs the RWTA network and SVPG, a new R-STDP learning method. In this thesis, SVPG is applied to both standard RL tasks and also a visual search scanpath modeling task. To distinguish between the two applications and to avoid excessive division of sections, I present these two applications separately from the method development chapter.

In this chapter, SVPG is applied to different benchmark RL tasks and compared to representative methods of other types. This chapter first introduces the task settings and alternative methods for comparison. Next is the empirical verification of the assumptions made in section 3.2.3 and 3.5.1. It also presents the results of the perturbation tests, ablation tests, and visualizations. The code for this part of the work is made publicly available at my code repository[1].

This chapter includes work that has been published in a jointly-authored publication [74]. Among the materials included in this chapter, the author of this thesis completed the text, experiments, visualizations, and analyses of the results. Shangqi Guo, Ying Fang, Zhaofei Yu, and Jian K. Liu contributed to the design of the experiments.

4.1 Tasks

I use five tasks in my experiments: reward-based MNIST classification [66], Gym Inverted-Pendulum [67], ViZDoom HealthGathering [183], AI2THOR navigation [69, 70], and robot-arm

[1] `https://github.com/yzlc080733/SVPG2023`

reaching (built on PyRep [184] and CoppeliaSim [71]). In the following texts and figures, I use MNIST, GYMIP, DOOM, AI2THOR, and ROBOTARM to refer to these tasks. Details of these tasks are described below.

- **MNIST**. In MNIST, the objective is to select the correct label given the image input, which are hand-written digits [66]. The state is a vector of length 784 reshaped from the image, the action space corresponds to the 10 labels, and the reward is $\{-1, +1\}$ corresponding to wrong or correct predictions. Each episode contains only one time step. The agent's observation is randomly selected from the MNIST dataset. The original images have a value range of $\{0, \dots, 255\}$, and are divided by 255 to be converted to the range of $[0, 1]$. The action space contains 10 actions, each corresponding to the 10 classes. The maximum length of training is set to 20k steps, and each step samples a batch of 100 images.
- **GYMIP**. In GYMIP, the objective is to balance a pendulum for as long as possible [67]. The maximum episode length is set to 200, the state is a length-4 vector of physical variables which are mapped to the range of $[0, 1]$, the action space is $\{-3, -1.5, 0, 1.5, 3\}$ deciding the force applied to the cart, and the reward is always $+1$. The episodes end early if the pendulum falls. Note that the original observation provided by the environment is a 4-dimensional vector with no predefined ranges. To normalize the observations, I use a random policy to sample from the environment and use the samples' range to determine a linear mapping to the range of $[0, 1]$. In the experiments, the sampled ranges are $[-0.4, 0.4]$, $[-0.2, 0.2]$, $[-1.7, 1.7]$, $[-1.25, 1.25]$. The maximum length of training is set to 2k episodes.
- **DOOM**. In DOOM, the objective is to navigate and pick up boxes to survive as long as possible [183]. In each episode, the game lasts a maximum of 2100 screen frames. I adopt the frame-skipping technique, i.e., repeat an action for a fixed number of frames. In this thesis, I choose to repeat each action for 4 frames, resulting in a maximum number of 525 time steps in each episode. The state is processed from the game screen, offering first-person visual observation of the game environment. The original game screen has a resolution of 320×240, which is transformed into grayscale, resized to 80×60, and reshaped to a vector with length 4800 to serve as the state observation. The action space contains 5 actions corresponding to the keyboard actions in the game: "move forward", "turn left", "turn right", "turn left while moving forward", and "turn right while moving forward". The rewards are determined by the player states, i.e., -50 when dead, $+10$ when picking up

a health kit, and +1 otherwise. An example of the original state observation is shown in Figure 4.1. The maximum length of training is set to 2k episodes for SVPG and BP, and 10k episodes for the method of backpropagation through time (BPTT), since it exhibits slower learning (introduction to the methods for comparison is below).

- **AI2THOR**. In AI2THOR, the objective is to navigate to a television in a realistic room. The agent needs to be within 1.5m distance from the television, and the television needs to be in the agent's view to mark a successful navigation. The room map is `FloorPlan_Train7_5` from the AI2THOR (specifically, RoboTHOR) platform [70]. The starting points are randomly selected from 38 randomly generated points (30 for training, 8 for validation/testing). The maximum episode length is set to 200. The episode ends early if the target object is found. The state observation is a view of the room from the agent's viewpoint and is an 80×60 RGB image. The parameters of the camera, e.g., angle of view, are the default ones provided by the AI2THOR platform. I convert the observed image to grayscale and reshape it to a 4800-length vector. Figure 4.2 presents an example of the original RGB image and the state observation after pre-processing. The action space consists of 5 actions: forward, turn left, turn right, move left, and move right. These actions are implemented by a list of operations provided by the AI2THOR platform. For example, turning left is achieved by `RotateLeft` and `MoveAhead`; moving left is achieved by `RotateLeft`, `MoveAhead`, and then `RotateRight`. The step sizes are 0.15m for movements and 90 degrees for rotations. The reward is defined to be: $+50$ for target-reaching, -5 for failure in action (e.g., collision), $+1$ for target-approaching, and -1 for target-deviation. The maximum length of training is set to 8k episodes.

- **ROBOTARM**. In ROBOTARM, the objective is to move the gripper of a robot arm to the cube on the table. The simulation environment is built upon the example Panda arm control scene from PyRep [184]. I add a vision sensor and a cube, and configure the initial pose of the robot arm. The scene is included in the code repository[2]. Each episode contains a maximum of 60 steps. The episode ends early if the target is reached. The state is a 64×64 RGB image obtained from a vision sensor attached at the end point of the arm. For pre-processing, I first convert the image to grayscale, then clip the pixel values to the range of $[0.4, 0.9]$, and finally linearly map it to the range of $[0, 1]$. Figure

[2]`https://github.com/yzlc080733/SVPG2023/tree/main/ROBOTARM/env_data`

4.3 presents an overview of the environment and an example of the original RGB image and the image after pre-processing. The input to the networks is a length-4096 vector reshaped from the processed image. The action space consists of 5 actions: move in 4 ways horizontally with a step size of 0.03m, and move upward/downward with a step size of 0.05m. The end-effector of the arm is constrained to move in a 0.2m $\times$ 0.3m $\times$ 0.22m rectangular space. The target object is a cube with a side length of 0.05m, and its position is randomly selected from 50 randomly generated positions (among which 45 for training and 5 for validation/testing). The reward is defined to be: +10 for target-reaching, +1 for target-approaching, and 0 otherwise. The maximum length of training is set to 10k episodes.

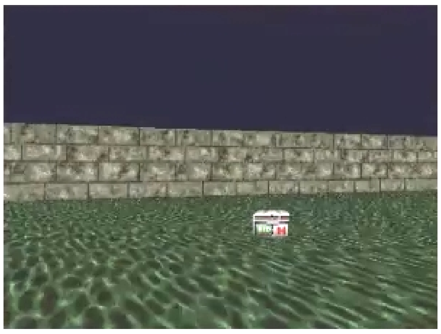


Figure 4.1: Example state original observation in the DOOM task.

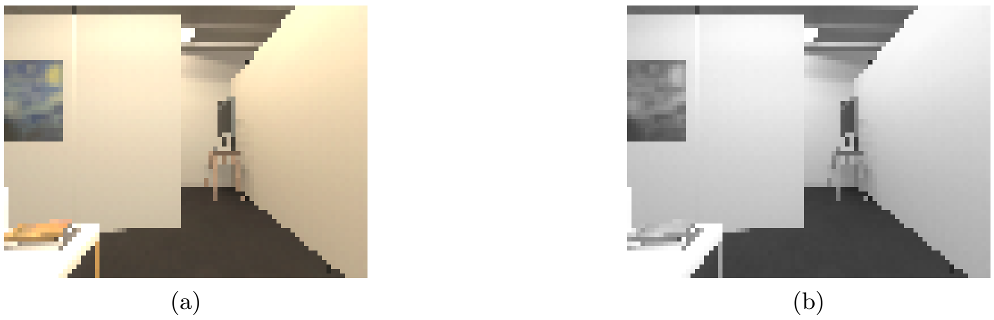


Figure 4.2: Example state observations in the AI2THOR task. (a) An original state observation. (b) A pre-processed state observation.

The motivations for selecting these tasks are as follows. (1) The MNIST task is selected because it is a single-step RL task, making its performance less affected by the training efficiency and exploration strategy. (2) The GYMIP task is selected because it is a standard task widely used in the literature [46, 185, 186]. Also, its state variables are unbounded, providing a good example of state mapping for SVPG. (3) The DOOM task is selected because it involves a high-dimensional vision input and a long episode horizon, which challenges the methods' overall

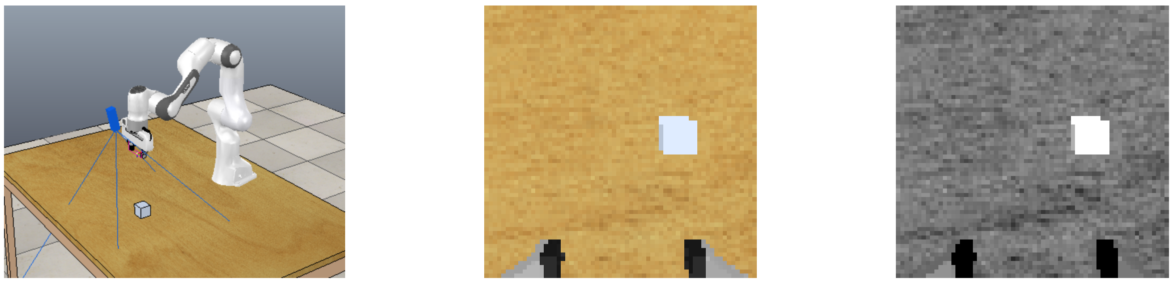


Figure 4.3: Example images in the ROBOTARM task. (a) An overview of the environment. The blue cuboid close to the gripper is the vision sensor. (b) An example state observation. (c) A pre-processed state observation.

capability. (4) The AI2THOR and ROBOTARM are used to reflect the methods' applicability to real-world tasks. In particular, AI2THOR provides photo-realistic scenes [69] (lighting, texture, etc.) and simulates collisions and noises in the robot's movements. In addition, the robot needs to generalize to new starting points (AI2THOR) or target positions (ROBOTARM).

I would like to emphasize that an encoding of the state observation is necessary to get the firing probabilities of the state neurons, because the latter is constrained to range $[0, 1]$. In MNIST, DOOM, AI2THOR, and ROBOTARM, the elements in state observations have limited values so can be linearly mapped to $[0, 1]$. In GYMIP, the state values are unbounded; therefore, pre-training samples are needed to estimate the range and then clip and map the observations to the range of $[0, 1]$.

4.2 Methods for Comparison

4.2.1 Method Selection

I select three representative learning methods for comparison. The first one serves as a conventional approach in deep RL and the other two are common approaches in SNN-based RL.

- *BP* [187] on a three-layer multi-layer perceptron (MLP) with the ReLU function for the hidden layers, which is a conventional baseline ANN model.
- *ANN2SNN* with the methods from [125] and code implementation from SpikingJelly [188]. This method is based on the training results from the BP method.
- Fast sigmoid *BPTT* (backpropagation through time) from [189]. The code implementation is based on snnTorch [190].

For these methods, the number of hidden layers is set to 1. In all the comparisons, I use the same optimizer (RMSprop or Adam), discount factor γ, number of hidden neurons, and base RL algorithm.

I also considered two local-learning-rule-based methods. The first is a hybrid method of STDP and R-STDP [191], implemented on SpykeTorch [192], and designed for MNIST. I refer to it as *Mozafari et al.* This method enables the training of deep networks by applying STDP to hidden layers. For a fair comparison, I changed its network structure to an MLP with the same shape as other methods and removed the difference of the Gaussian filter. Other designs, including latency encoding and adaptive learning rate, are kept. The second is a local gradient-based optimization method [193], implemented for CartPole, an environment similar to GYMIP. I refer to it as *Aenugu et al.* This method uses the generalized linear model as neurons and updates connection weights based only on the local spiking activity and the global reward information. I reduced the size of the hidden layer to make the comparison fair. The input encoding, sparse connection, critic model, and voting mechanism are kept. Note that this method only considers networks with one hidden layer. Therefore, my setting of the number of hidden layers in other methods to one enables a fair comparison to this method.

Notice that the RWTA network in my method is fully connected, so it has more learnable parameters than other methods under the same number of hidden neurons. Therefore, I add a variant *SVPG-shrink* with fewer hidden WTA circuits, of which the number of learnable parameters is equal to or less than the networks in other methods.

4.2.2 Implementation Details

The performance of the methods can be dependent on the network sizes and RL hyperparameters. For the networks, I adapt the methods to have networks of the same size. Considering that different methods may need different RL hyperparameters to generate the best performance, I train the methods with a range of RL hyperparameters and report the result with the best zero-perturbation testing performance. Note that the methods of “Mozafari et al” and “Aenugu et al” have their own design of optimization, so the results are based on their original setting.

RL hyperparameters

For all the tasks in my experiments, the discount factor γ is set to 0.97. This makes the discounted return reflect the length of the episodes in GYMIP and DOOM, so that the agent learns to complete the task. In MNIST, DOOM, AI2THOR, and ROBOTARM, I use Monte-Carlo sampling to learn the critic; in GYMIP, I use temporal difference to learn the critic and adopt a memory buffer with size 1000. The epoch number in PPO is set to 5, and the clipping parameter is set to 0.2. Due to the instability in training BPTT on DOOM, I specially set the epoch number to 10 for BPTT on DOOM. To encourage the agent to explore different actions, I use a weighted sum of the policy gradient and the entropy of the agent's policy distribution to train the agent. This introduces a hyperparameter of the entropy ratio.

SVPG		Entropy Ratio							
		0	0.1	0.2	0.5	1	2	5	10
Learning Rate	0.01	0.7448	0.8056	0.8042	0.8914	0.8605	0.8894	0.4123	0.3054
	0.001	0.9284	0.9270	**0.9292**	0.9271	0.9270	0.9262	0.8467	0.7938
	0.0001	0.9233	0.9236	0.9232	0.9230	0.9235	0.9231	0.9088	0.8997

(a)

SVPG-shrink		Entropy Ratio							
		0	0.1	0.2	0.5	1	2	5	10
Learning Rate	0.01	0.8053	0.8344	0.8960	0.9211	0.8598	0.9139	0.4226	0.3295
	0.001	0.9267	0.9281	**0.9282**	0.9269	0.9261	0.9242	0.8491	0.8018
	0.0001	0.9235	0.9246	0.9237	0.9243	0.9252	0.9232	0.9157	0.9036

(b)

BP		Entropy Ratio							
		0	0.1	0.2	0.5	1	2	5	10
Learning Rate	0.01	0.5871	0.9339	0.9357	0.8858	0.5548	0.2746	0.1121	0.1093
	0.001	0.9781	0.9763	0.9691	0.9565	0.9512	0.9276	0.9069	0.8924
	0.0001	0.9760	0.9761	0.9774	0.9779	**0.9792**	0.9739	0.9639	0.9578

(c)

BPTT		Entropy Ratio							
		0	0.1	0.2	0.5	1	2	5	10
Learning Rate	0.01	0.4356	0.4257	0.5191	0.1884	0.1032	0.1032	0.1695	0.1370
	0.001	0.7783	0.8371	0.8580	0.9001	0.9252	0.4060	0.2116	0.1638
	0.0001	0.9729	0.9735	0.9744	**0.9748**	0.9745	0.9624	0.9434	0.9211

(d)

ANN2SNN		Entropy Ratio							
		0	0.1	0.2	0.5	1	2	5	10
Learning Rate	0.01	0.5866	0.9349	0.9292	0.8805	0.5477	0.2680	0.1121	0.1093
	0.001	0.9788	0.9764	0.9660	0.9525	0.9490	0.9253	0.9040	0.8899
	0.0001	0.9765	0.9766	0.9779	0.9782	**0.9784**	0.9727	0.9618	0.9550

(e)

Figure 4.4: The effect of hyperparameters of learning rate and entropy ratio in the MNIST task with different models (a: SVPG; b: SVPG-shrink; c: BP; d: BPTT; e: ANN2SNN). Values shown are the classification accuracies (higher is better).

SVPG		Entropy Ratio							
		0	0.1	0.2	0.5	1	2	5	10
Learning Rate	0.01	96.58	149.96	112.58	173.95	184.05	**200.00**	183.24	165.09
	0.001	166.17	143.15	197.29	182.36	**200.00**	**200.00**	**200.00**	**200.00**
	0.0001	62.56	88.81	132.89	149.40	148.90	161.22	140.45	137.65

(a)

BP		Entropy Ratio							
		0	0.1	0.2	0.5	1	2	5	10
Learning Rate	0.01	105.02	**200.00**	**200.00**	**200.00**	**200.00**	**200.00**	**198.99**	183.77
	0.001	99.71	**198.52**	**200.00**	**200.00**	**200.00**	**200.00**	**199.56**	191.98
	0.0001	**199.97**	**200.00**	**200.00**	**200.00**	**200.00**	**200.00**	**199.79**	146.59

(b)

BPTT		Entropy Ratio							
		0	0.1	0.2	0.5	1	2	5	10
Learning Rate	0.01	128.25	128.14	92.38	119.06	121.89	139.57	165.00	103.20
	0.001	178.22	137.88	141.44	159.98	143.33	174.51	167.23	120.84
	0.0001	141.44	176.86	129.18	191.03	159.71	192.64	**198.18**	113.56

(c)

ANN2SNN		Entropy Ratio							
		0	0.1	0.2	0.5	1	2	5	10
Learning Rate	0.01	104.45	188.40	187.12	172.05	189.72	**199.83**	181.75	150.33
	0.001	99.70	176.54	175.77	191.43	189.83	197.75	171.04	157.82
	0.0001	187.74	190.20	**200.00**	187.35	**198.34**	179.52	167.63	99.39

(d)

Figure 4.5: The effect of hyperparameters of learning rate and entropy ratio in the GYMIP task with different models (a: SVPG; b: BP; c: BPTT; d: ANN2SNN). Values shown are episode lengths (higher is better).

In practice, I find that the learning rate and the entropy ratio have a large impact on the zero-noise test results. Therefore, I tune these two hyperparameters for each method and report the one with the best testing performance. On MNIST and GYMIP, I compare the entropy ratio with values $\{0, 0.1, 0.2, 0.5, 1, 2, 5, 10\}$ and the learning rate with values $\{0.01, 0.001, 0.0001\}$. On DOOM, I compare entropy ratio with values $\{0.02, 0.2, 2, 5\}$ and learning rates $\{0.001, 0.0001\}$. On AI2THOR, I compare the entropy ratio with values $\{0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5\}$ and learning rates $\{0.0001, 0.00001\}$. On ROBOTHOR, I compare the entropy ratio with values $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10\}$ and learning rates $\{0.0001, 0.00001\}$. For each setting of the hyperparameters, I perform 10 independent trainings for GYMIP and DOOM, and 3 independent trainings for MNIST, AI2THOR, and ROBOTARM. The tuning results are provided in Figure 4.4, 4.5, 4.6, 4.7, and 4.8. The values presented are the mean values of the zero-perturbation testing performances.

Note that some methods may produce performances close to their best one under multiple different RL hyperparameters. This is particularly evident in GYMIP and AI2THOR. Therefore,

SVPG		Entropy Ratio			
		0.02	0.2	2	5
Learning Rate	0.001	503.24	**525.00**	**525.00**	462.67
	0.0001	435.57	513.73	517.78	483.61

(a)

BP		Entropy Ratio			
		0.02	0.2	2	5
Learning Rate	0.001	**525.00**	224.25	129.12	124.32
	0.0001	**525.00**	517.14	292.26	217.13

(b)

BPTT		Entropy Ratio			
		0.02	0.2	2	5
Learning Rate	0.001	121.92	124.91	108.34	97.14
	0.0001	**394.98**	296.02	112.10	127.80

(c)

ANN2SNN		Entropy Ratio			
		0.02	0.2	2	5
Learning Rate	0.001	**525.00**	200.85	125.76	122.96
	0.0001	475.15	495.38	283.69	223.23

(d)

Figure 4.6: The effect of hyperparameters of learning rate and entropy ratio in the DOOM task with different models (a: SVPG; b: BP; c: BPTT; d: ANN2SNN). Values shown are episode lengths (higher is better).

for the robustness test later, I choose to average the performances over satisfactory results. Specifically, on GYMIP, I consider the average of all hyperparameters that generate the best zero-noise performance; to tolerate noises in the zero-noise performance, I adopt the hyperparameters with zero-noise performance greater than or equal to 99% of the best performance. On AI2THOR, BP and ANN2SNN generate multiple best zero-noise results; all the corresponding hyperparameters are considered. For the other tasks, i.e., MNIST, DOOM, and ROBOTARM, there is generally only one best performance, so the hyperparameters corresponding to the best performance are selected. For the occasional multiple best performances (DOOM), the performance at a level of perturbation is considered. For input perturbations, the level value is set to 0.2; for network parameter perturbations, the level value is 2.0. Based on this criterion, the BP-0.001-0.02 (method-learning rate-entropy ratio) is selected; the SVPG-0.001-2.0 is selected for input perturbations and the SVPG-0.001-0.2 is selected for network parameter perturbations.

During training, a checkpoint of the network parameters is saved every 100 episodes. The checkpoints are validated using the validation environment, and the best one is used in testing. The validation environment in the MNIST task is created by randomly dividing the training set according to a ratio of 9:1, where the latter part is used as the validation set. The validation environments in GYMIP and DOOM are the same as the training environments. For AI2THOR, the validation environment uses the same scene as in training, but the starting points are randomly sampled from a list of positions different from training. For ROBOTARM, in the validation environment, the initial position of the target cube is randomly sampled from a list different from training.

SVPG		Entropy Ratio							
		0.02	0.05	0.1	0.2	0.5	1	2	5
Learning	0.0001	200.00	142.13	142.15	200.00	200.00	146.04	83.04	200.00
Rate	0.00001	**24.00**	82.67	82.67	44.83	27.67	33.67	37.38	32.38

(a)

BP		Entropy Ratio							
		0.02	0.05	0.1	0.2	0.5	1	2	5
Learning	0.0001	34.67	28.60	40.13	28.77	**24.00**	82.69	200.00	141.79
Rate	0.00001	38.42	32.81	25.63	26.33	**24.00**	24.02	24.04	24.06

(b)

BPTT		Entropy Ratio							
		0.02	0.05	0.1	0.2	0.5	1	2	5
Learning	0.0001	200.00	200.00	200.00	200.00	200.00	200.00	200.00	200.00
Rate	0.00001	200.00	200.00	**82.77**	141.33	200.00	200.00	141.33	141.73

(c)

ANN2SNN		Entropy Ratio							
		0.02	0.05	0.1	0.2	0.5	1	2	5
Learning	0.0001	87.42	66.73	84.79	45.98	24.02	83.65	200.00	141.63
Rate	0.00001	141.58	40.90	32.69	62.21	**24.00**	24.13	24.15	**24.00**

(d)

Figure 4.7: The effect of hyperparameters of learning rate and entropy ratio in the AI2THOR task with different models (a: SVPG; b: BP; c: BPTT; d: ANN2SNN). Values shown are average numbers of steps (lower is better).

4.3 Results

4.3.1 Assumption Verification

In the theory part (chapter 3), there are two assumptions. (1) In section 3.2.3, I assume that iterating the firing probabilities with the policy inference function Eq. (3.5) can reach numeric convergence. (2) In section "practical considerations" 3.5.1, I propose an approximated implementation of SVPG. Here I use empirical results to verify these assumptions.

Convergence verification

The policy inference part of SVPG, either rate-based or spike-based, relies on the iteration of the policy inference function Eq. (3.5). Here I use the MNIST task to empirically verify the convergence of the iteration process.

I monitor the firing probabilities of hidden and action neurons and set a stopping criterion for the iteration, which is that the mean absolute changes in those probabilities in an iteration are smaller than 0.005, or that the iteration exceeds 50 steps. Note that this stopping criterion is also adopted in training rate-based SVPG. I feed the RWTA network with the testing images

SVPG		Entropy Ratio									
		0.01	0.02	0.05	0.1	0.2	0.5	1	2	5	10
Learning	0.0001	31.50	24.87	35.20	41.93	38.40	38.33	45.47	12.07	10.47	19.83
Rate	0.00001	16.60	10.93	8.47	11.47	10.17	8.20	**7.43**	12.90	9.07	7.47

(a)

BP		Entropy Ratio									
		0.01	0.02	0.05	0.1	0.2	0.5	1	2	5	10
Learning	0.0001	32.20	42.47	60.00	52.87	46.00	56.60	60.00	60.00	60.00	60.00
Rate	0.00001	7.00	7.53	7.33	7.13	7.13	7.13	**6.93**	7.40	7.13	7.67

(b)

BPTT		Entropy Ratio									
		0.01	0.02	0.05	0.1	0.2	0.5	1	2	5	10
Learning	0.0001	60.00	60.00	60.00	60.00	60.00	60.00	60.00	60.00	60.00	60.00
Rate	0.00001	60.00	27.87	31.53	52.73	**25.40**	42.47	31.80	60.00	60.00	60.00

(c)

ANN2SNN		Entropy Ratio									
		0.01	0.02	0.05	0.1	0.2	0.5	1	2	5	10
Learning	0.0001	39.07	42.47	60.00	52.87	46.40	58.07	60.00	60.00	60.00	60.00
Rate	0.00001	27.40	20.53	16.93	20.20	20.67	23.93	17.13	**7.20**	10.27	14.00

(d)

Figure 4.8: The effect of hyperparameters of learning rate and entropy ratio in the ROBOTARM task with different models (a: SVPG; b: BP; c: BPTT; d: ANN2SNN). Values shown are average numbers of steps (lower is better).

one by one and record their corresponding iteration lengths. The distribution of the iteration lengths is plotted in Figure 4.9. As shown, for all the tested images, the iteration converges within 30 steps. Also, most images correspond to an iteration length smaller than 10. These empirically verify that the policy inference function converges under most input cases.

Rate-based SVPG implementation

I compare the rate-based SVPG implementation with the original spike-based implementation on MNIST and GYMIP. The testing results with different strengths of perturbations to input and network parameters are shown in Figure 4.10 and Figure 4.11. The curves are averaged across 10 independent trainings, and the shaded regions represent the standard deviation values. Details of perturbations are in Section 4.3.3.

As shown, the two implementations generate similar results under the tested perturbations, indicating that the rate-based implementation can be used as a replacement for spike train simulation. In the following experiments, I use the rate-based implementation to represent SVPG.

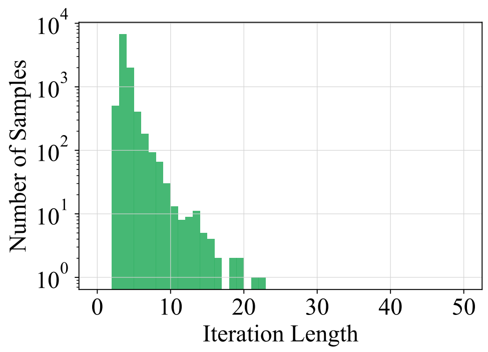


Figure 4.9: Distribution of SVPG convergence iteration lengths in the MNIST task.

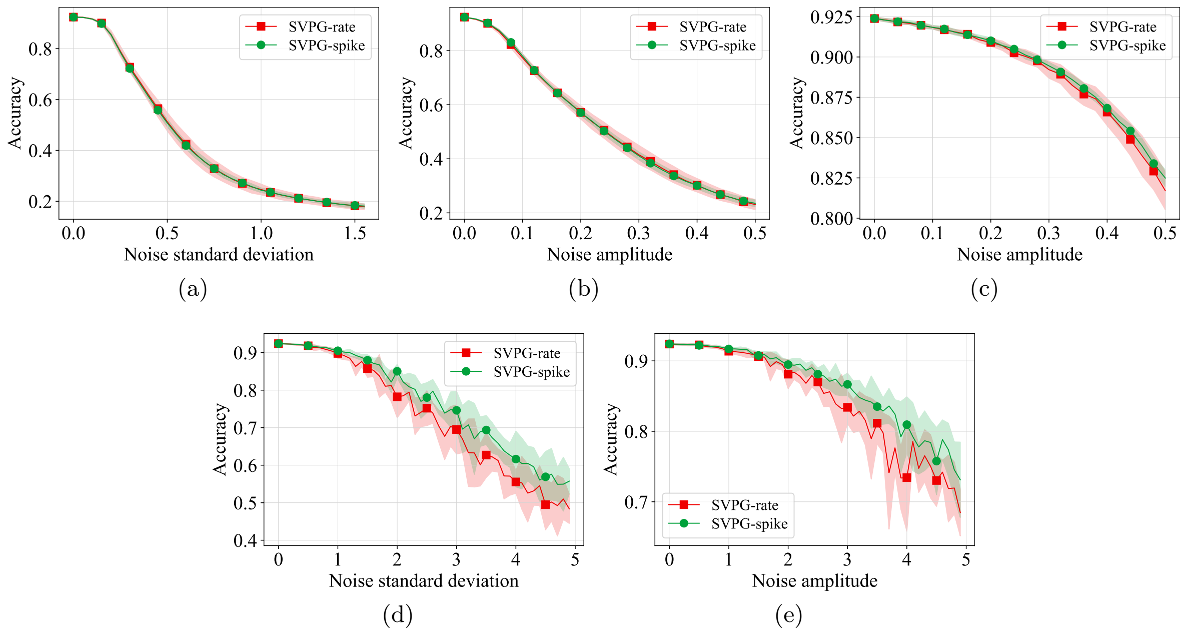


Figure 4.10: Comparison of spike-based and rate-based SVPG implementations in the MNIST task in different scenarios of noise perturbation. (a) Input Gaussian noise. (b) Input salt noise. (c) Input pepper noise. (d) Network Gaussian noise. (e) Network uniform noise.

4.3.2 Task Performances

I train different methods respectively on the five tasks. For the optimizer, I use Adam on MNIST, DOOM, AI2THOR, and ROBOTARM, and use RMSprop on GYMIP. This brings variances to the selection of the optimizer and can better check the effectiveness of SVPG. For the base RL algorithm, in a preliminary version of this study [194], I used REINFORCE. Here I upgrade it to PPO-clip because of its popularity and training efficiency.

Zero-perturbation testing performances

The zero-perturbation testing performances are shown in Table 4.1.

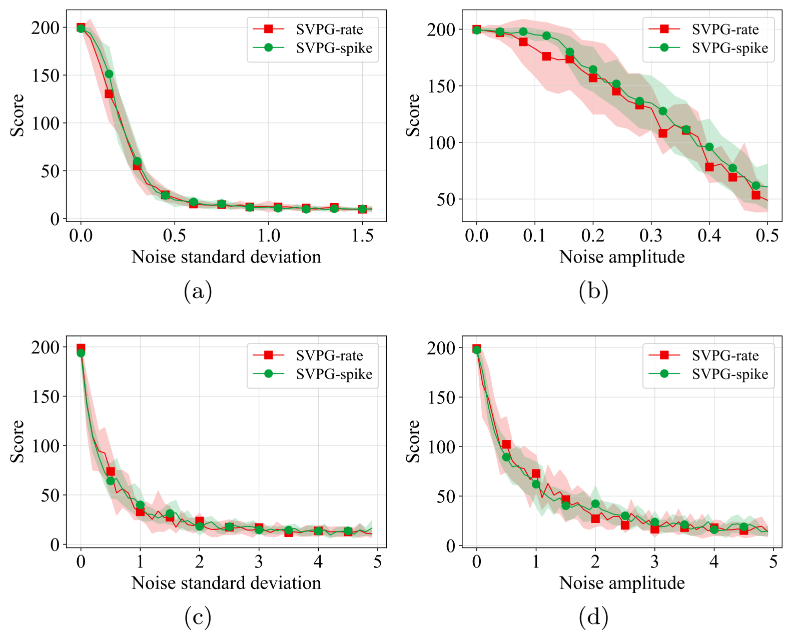


Figure 4.11: Comparison of spike-based and rate-based SVPG implementations in the GYMIP task with different scenarios of noise perturbation.. (a) Input Gaussian noise. (b) Input uniform noise. (c) Network Gaussian noise. (d) Network uniform noise.

Table 4.1: Zero-Noise Testing Performances on the 5 Tasks. △: Higher better. ▽: Lower better.

Tasks	SVPG	BP	BPTT	ANN2SNN	Mozafari et al.	Aenugu et al.
MNIST △	0.929±0.001	0.979±0.001	0.975±0.001	0.978±0.002	0.587±0.010	-
GYMIP △	200.00±0.00	200.00±0.00	198.18±3.68	200.00±0.00	-	195.11±7.30
DOOM △	525.00±0.00	525.00±0.00	394.98±151.14	525.00±0.00	-	-
AI2THOR ▽	24.00±0.00	24.00±0.00	82.77±82.89	24.00±0.00	-	-
ROBOTARM ▽	7.43±0.26	6.93±0.19	25.40±24.47	7.20±0.16	-	-

The values for MNIST, GYMIP, and DOOM are from 10 independent trainings and the values for AI2THOR and ROBOTARM are from 3 independent trainings. The presented values are in the form of mean±standard deviation. For the first three tasks, a higher performance value is better; for the last two tasks, a lower performance value is better.

- On MNIST, the performance is measured by the testing accuracy. SVPG performs not as well as BP, BPTT, and ANN2SNN.
- On GYMIP, the performance is measured by the length of testing episodes, and the optimum value is 200. SVPG achieves optimal performance.
- On DOOM, the performance is measured by the length of testing episodes, and the optimal value is 525. SVPG achieves optimal performance.

- On AI2THOR, the performance is measured by the average number of steps the agent used to reach the target from different starting points. The optimal value is 24. SVPG achieves optimal performance.
- On ROBOTARM, the performance is measured by the average number of steps the agent used to reach the target from different starting points. SVPG achieves near-optimal performance.

These results indicate that SVPG with the RWTA network is able to solve image-input, long-horizon, and realistic RL tasks. The results also show that SVPG has a better performance than the compared R-STDP or local rule-based methods, i.e., Mozafari et al. and Aenugu et al.. There are two contributors to these results. The first is that the RWTA network brings better capacity than the layered networks. The recurrent design and the extra connections between hidden circuits and between state and action neurons allow representation of a more complex mapping. The contributions of the connections will be demonstrated in the ablation tests (section 4.3.4). The second is that the variational inference-based approximation in the policy inference enables the SVPG learning rules to be applied to all the learnable parameters, instead of only the output layer as in the Mozafari et al. method.

Computational costs

The computational costs are important for the practical applications of the methods. Here I provide results on the time complexity, space complexity, and sample efficiency. The *Mozafari et al.* R-STDP method and the *Aenugu et al.* method are not measured because their implementation [191, 193] does not support parallel processing of multiple samples, resulting in low space complexity and high time complexity.

(1) Time complexity. I measure the time required for the inference and optimization steps to reflect the time complexity. The MNIST task is selected for this test because it has a consistent batch size and that the high-dimensional state induces a large computational cost. The implementation of the methods is all based on PyTorch and runs on the same machine, with a NVIDIA T600 GPU. The results are shown in Table 4.2. The values are averaged across 500 inference/optimization steps.

As shown, the rate coding variant of SVPG is more than 100 times faster than the spiking variant, which supports that using the rate approximation can reduce computation costs. Since

Table 4.2: Time Complexity on MNIST. (Time measured in milliseconds)

Time	SVPG-rate	SVPG-spike	BP	BPTT
Inference	3.26±0.28	677.21±13.50	**0.18±0.02**	8.51±0.17
Optimization	**1.40±0.17**	1.49±0.15	1.43±0.06	5.54±0.06

the inference stage of RWTA requires iterations, the cost is higher than BP in which the MLP only needs a forward propagation. In addition, the rate-based SVPG is faster than the BPTT method. For the optimization stage, SVPG is faster than all the compared methods. This is because SVPG is a local learning method. That is, the update of the parameters does not need a layer-by-layer computation process and can be completed in one step. Future work could leverage the local learning property of SVPG and implement it with neuromorphic hardware to improve the inference speed.

(2) Space complexity. I measure the memory costs of each method to reflect their space complexity. Again, the MNIST task is selected because it offers a consistent episode length that enables fair comparison. Different from normal training, in this test, I set the program to only use one CPU thread and no GPU, so the memory usage includes all the variables a method creates. I use the `psutil` package for Python to get the memory usage of the program before initialization and after training for 20 episodes, and use their difference as the memory costs. The results are presented in Table 4.3. The values are the mean and standard deviation values of results from 10 independent runs.

Table 4.3: Space Complexity on MNIST. (Total memory used in megabytes)

Method	SVPG-rate	SVPG-spike	BP	BPTT
Memory Cost	603.7±10.8	611.2±17.7	520.4±6.5	1180.5±27.5

As shown, the memory cost of the spiking variant of SVPG is slightly higher than that of the rate-based variant. This is mainly due to the extra spike train recording in policy inference. Nevertheless, the memory cost of the spike-based SVPG is only slightly higher than BP (17.4%) and much lower than BPTT (48.2%). The similarity to BP can be attributed to the fact that SVPG only needs to store the final state of the network for the optimization process, which is similar to that of BP. The slightly higher cost in SVPG could probably be due to the experiment scripts not being sufficiently optimized. As for BPTT, its memory cost depends on the implementation in the snnTorch library [190], which may not be optimal. Nevertheless, the

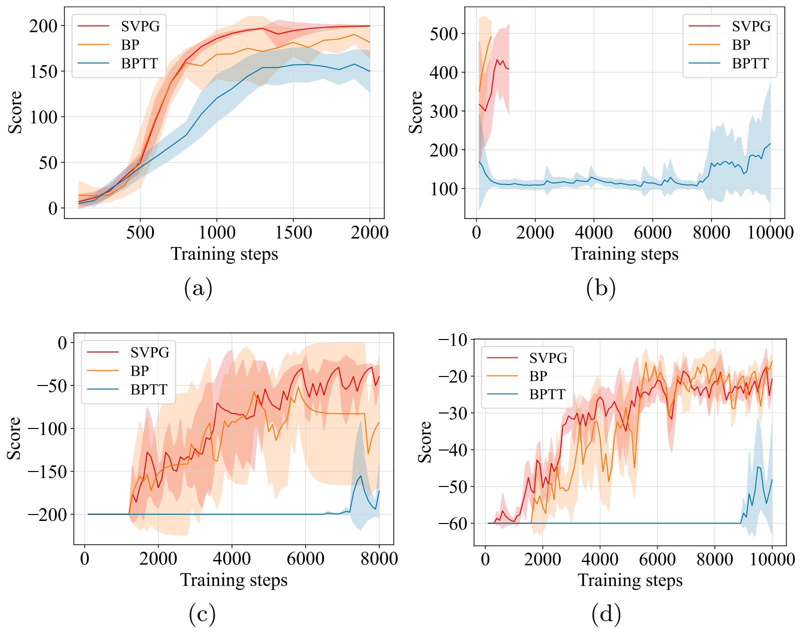


Figure 4.12: Learning curves of different models in four tasks. (a) GYMIP. (b) DOOM. (c) AI2THOR. (d) ROBOTARM. The horizontal axis is the RL training step number. The vertical axis is the validation score/performance (higher is better), which is measured during training in periods of 100 training steps. The curves are smoothed using exponential smoothing [195] with smoothing factor $\alpha = 0.4$ ($0 < \alpha < 1$; a smaller α means stronger smoothing).

results indicate that SVPG may not cause much difficulty for practical applications in terms of space complexity.

(3) RL sample efficiency. The total training time depends on both the time complexity (time per step) and the sample complexity (number of steps in RL training). Here I plot the learning curves to compare the RL sample efficiency of the methods. The four tasks with sequential decision-making are used (i.e., GYMIP, DOOM, AI2THOR, and ROBOTARM). For each method and each task, a hyperparameter (entropy ratio and learning rate) that produces the best testing performance is chosen. The results are presented in Figure 4.12. The curves are the mean values of validation performances, and the shaded regions are the standard deviations. For MNIST, GYMIP, and DOOM, the curves are the average of 10 independent trainings. For AI2THOR and ROBOTARM, the curves are the average of 3 independent trainings.

Note that, for ease of viewing, I take the opposite value of the curves for AI2THOR and ROBOTARM. Specifically, the "score" equals (-1) times the step number. This makes the trends of these curves the same as those of the other three tasks. This is also done for other figures on these two tasks.

As shown, the curves of SVPG are generally close to those of BP. Both SVPG and BP learn steadily and faster than BPTT. This indicates that the sample efficiency of SVPG is similar to BP and better than BPTT. SVPG has the potential to be applied to scenarios where the efficiency of a traditional ANN is acceptable. A possible explanation of this result is that BPTT uses a surrogate gradient and has a longer path for gradient propagation than BP and SVPG. The approximation error in the gradient and the gradient explosion or vanishing effects make BPTT unstable. From this point, the sample efficiency could be an advantage of local learning rules like SVPG.

4.3.3 Perturbation Tests

It has been shown in many studies that SNNs (trained with ANN2SNN and BPTT methods) can exhibit better robustness to input and synapse weight noises [31,72] and adversarial attacks [196] than ANNs. Here I perform tests on trained models to investigate whether SVPG can produce robustness.

I test three types of perturbations, namely input noise, network parameter noise, and environmental variation (in GYMIP). The input noises reflect inevitable sensor noises in the real world. The network parameter noise corresponds to parameter inaccuracies in neuromorphic hardware [197]. The environment variation in GYMIP represents differences in environment dynamics between the simulation and the real world.

- **Input noise** is independently added to each dimension of state observations. For MNIST, DOOM, AI2THOR, and ROBOTARM, Gaussian, salt, salt&pepper, and Gaussian&salt noises are considered. For GYMIP, Gaussian and uniform noises are considered. Some illustrations of the MNIST and DOOM tasks are shown in Figure 4.13, 4.14.

- **Network parameter noise** is independently added to each learnable parameter in the policy networks. Gaussian and uniform noises are adopted. Considering that different parts of the trained networks may have different scales of parameters, I divide the parameters into groups and normalize the noise using the mean absolute values within groups. For the RWTA network, the synapse weights $\boldsymbol{W}$ are divided according to the types of neurons connected, e.g., connections between state neurons and action neurons; the intrinsic excitability $\boldsymbol{b}$ forms one group. For the layered networks in other compared methods, the parameters are divided by layers and weight/bias. Note that this type of noise is regenerated for each testing episode.

- **Environmental variations in GYMIP**. In the GYMIP task, the optimal policy relies on the length and thickness of the pendulum. In training, I set ⟨length=1.5, thickness=0.05⟩. In testing, I change the length to be in the range of $[0.5, 4.9]$ and the thickness of $[0.02, 0.30]$. Figure 4.15 illustrates these variations.

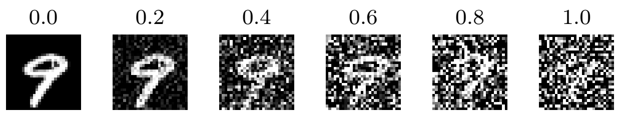


Figure 4.13: Example input images with different strengths of Gaussian noises in the MNIST task. Standard deviation noted above images.

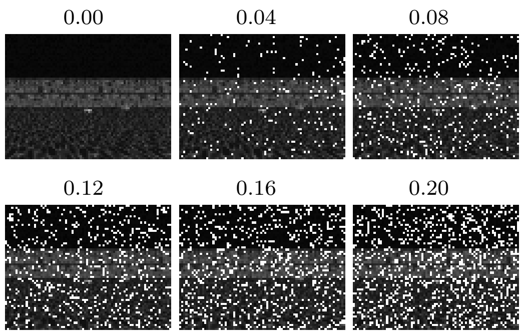


Figure 4.14: Example input images with different strengths of salt noises in the DOOM task. Noise ratio noted above images.

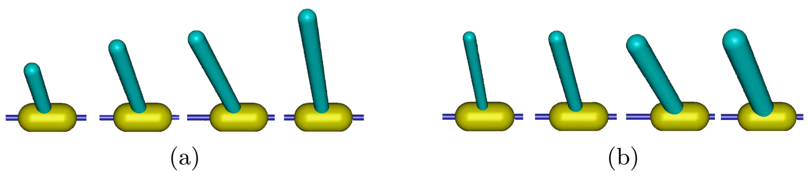
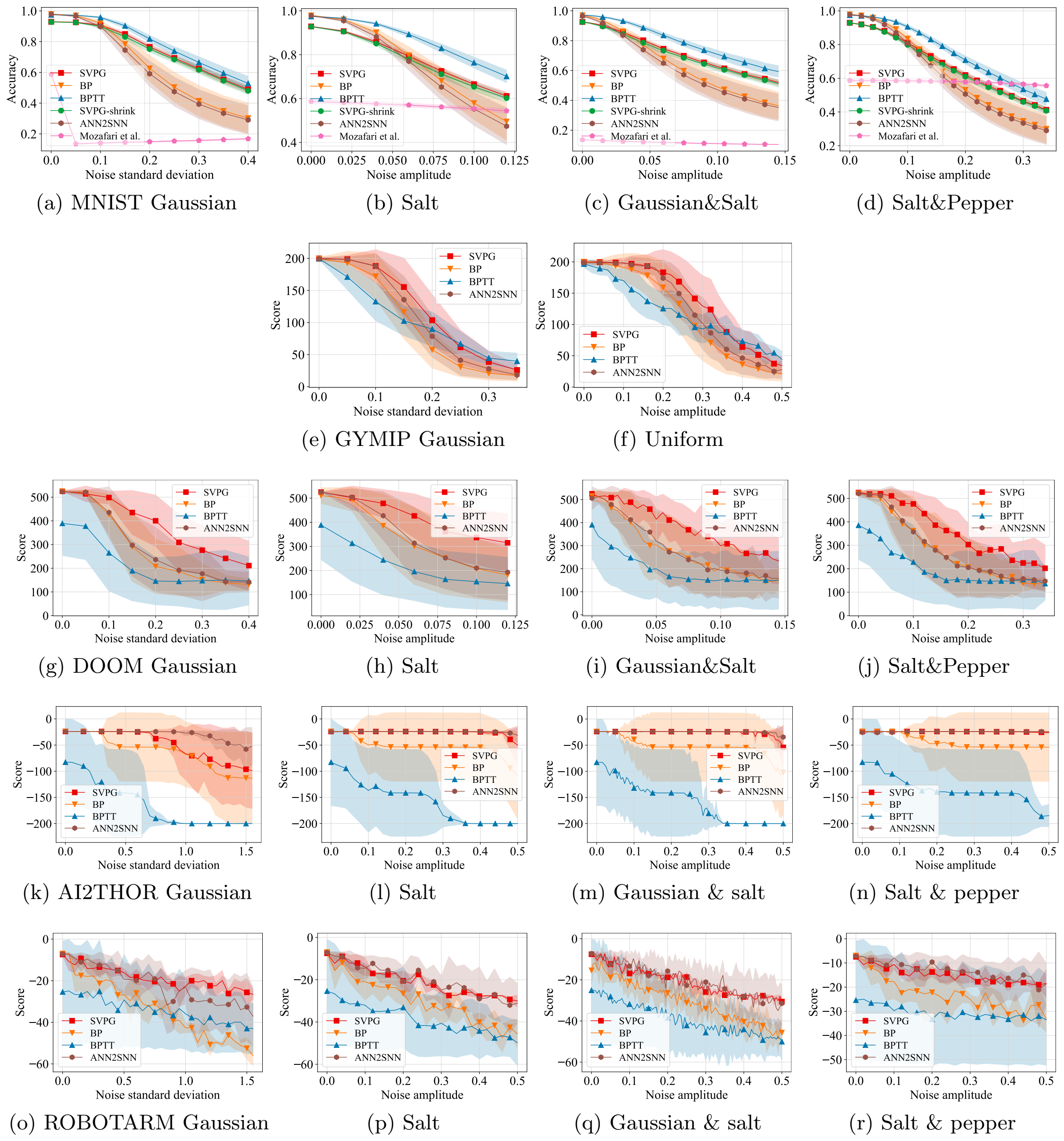


Figure 4.15: Environmental variations in GYMIP testing. (a) Length. (b) Thickness.

As mentioned above in section 4.2.2, for each type of perturbation and each method, I select the hyperparameter (learning rate and entropy ratio) that generates the best zero-noise performance. When there are multiple hyperparameters that are selected, the results are averaged to measure the robustness performance. In the following results, Figure 4.16, 4.17 and 4.18, the curves are from the average of 10 independent trainings for MNIST, GYMIP, and DOOM, and 3 independent trainings for AI2THOR and ROBOTARM. The shaded regions represent the standard deviation values. Note that the *Aenugu et al.* method contains an input encoder and a voting mechanism, which are not included in other methods. These may affect the robustness to input and network parameter perturbations. Therefore, I only test its robustness to environment variations in the GYMIP task.



(a) MNIST Gaussian (b) Salt (c) Gaussian&Salt (d) Salt&Pepper

(e) GYMIP Gaussian (f) Uniform

(g) DOOM Gaussian (h) Salt (i) Gaussian&Salt (j) Salt&Pepper

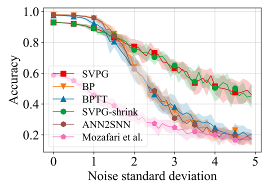
(k) AI2THOR Gaussian (l) Salt (m) Gaussian & salt (n) Salt & pepper

(o) ROBOTARM Gaussian (p) Salt (q) Gaussian & salt (r) Salt & pepper

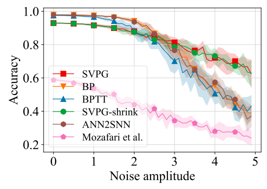
Figure 4.16: Input noise tests in the five tasks. (a)–(d) MNIST. (e)–(f) GYMIP. (g)–(j) DOOM. (k)–(n) AI2THOR. (o)–(r) ROBOTARM.

Input noises

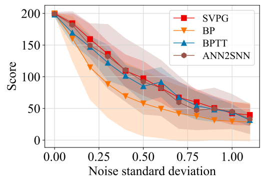
The results are plotted in Figure 4.16. In each test, I apply a range of different strengths of noise to the state inputs and test the agents' performance. For MNIST, the test uses all the samples from the testing images in the MNIST dataset. In GYMIP, DOOM, AI2THOR,



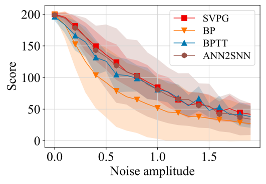
(a) MNIST Gaussian



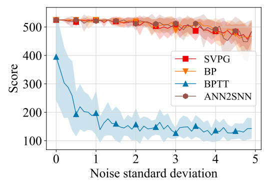
(b) MNIST Uniform



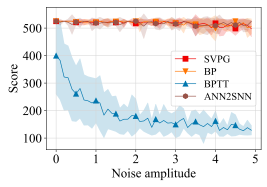
(c) GYMIP Gaussian



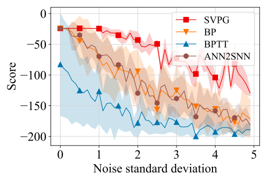
(d) GYMIP Uniform



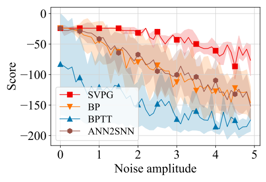
(e) DOOM Gaussian



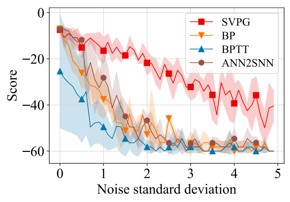
(f) DOOM Uniform



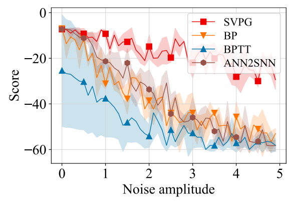
(g) AI2THOR Gaussian



(h) AI2THOR Uniform



(i) ROBOTARM Gaussian



(j) ROBOTARM Uniform

Figure 4.17: Network parameter noise tests in the five tasks.

and ROBOTARM, the test score for each training is the average value of 10 episodes. As the strength of the noises increases, the performance decreases. The speed of the decrease reflects the robustness of the methods.

As shown, for the tested types of input noises on all three tasks, the performance of SVPG generally degrades more slowly than other methods. Instead of being robust on one task and sensitive on another (such as BPTT which is the best on MNIST but the worst on GYMIP), SVPG displays a more consistent robustness across tasks. This shows that SVPG produces better inherent robustness to the tested input noises.

Network parameter noises

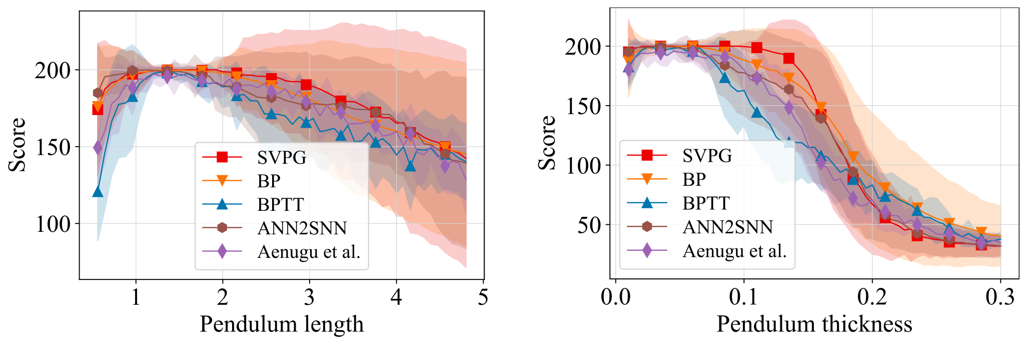
The results are plotted in Figure 4.17. The experiment settings are the same as the test on input noises.

As shown, on all 5 tasks, SVPG achieves the slowest degradation of performance as the amplitude of noise increases. These indicate that SVPG generally produces better robustness to network perturbations than the other methods.

Also note that in the above results on input and network parameter noises, *SVPG-shrink* and *SVPG* exhibit similar performances and robustness. This indicates that it is not the larger number of learnable parameters in the RWTA network that brings the difference in robustness.

Environmental variations in GYMIP

Results on variations in the pendulum's length and thickness in GYMIP are shown in Figure 4.18. When the shape of the pendulum deviates from the one in training, i.e., ⟨length=1.5, thickness=0.05⟩, the performance of all the methods degrades. For pendulum length, the performance of SVPG degrades more slowly than other methods. For pendulum thickness, SVPG is close to the best when the thickness is smaller than 0.15. These reveal that the policy trained using SVPG naturally adapts to a larger range of pendulums with different shapes.



(a) Pendulum length (b) Pendulum thickness

Figure 4.18: Environmental variations on GYMIP.

So far, I have tested the robustness to input noises, network parameter noises, and environmental variations. As discussed, SVPG shows better robustness to most types of perturbations than the compared methods. I emphasize that in all these robustness tests, the noises are only added in testing. The results support the idea that SNNs could have better inherent robustness than conventional neural network models.

A possible explanation of the robustness results on input and network parameters is that the simulation of the RWTA network involves a random firing process for each neuron, which serves as perturbations added to the training process. Therefore, after training, the policy can adapt to perturbations to the input, hidden, and action neurons, which correspond to input and network parameter perturbations. Additionally, as will be introduced later (section 4.3.5), the RWTA network has noisy hidden WTA circuits after training, which may also contribute to the robustness. The investigation of how the robustness is produced is left as future work.

4.3.4 Ablation Studies

I perform ablation tests to understand the effects of different parts of the connections in the RWTA network. The MNIST task is used because the classification accuracy can better reflect the differences between performances.

Effect of regional connection removal in training

In this test, I remove different parts of connections in the RWTA network before training to investigate their contributions to learning. I divide the connections in the network according to the types of neurons they connect – connections between hidden and hidden neurons, hidden and action, state and action, and state and hidden. For conciseness, I use HH, HA, SA, and SH to represent these types. For example, I use "RM-HH" to refer to removing all connections between hidden WTA circuits. By "original", I refer to the original RWTA network. I add perturbations to the input or network parameters during testing to better discriminate the performances. The results are shown in Figure 4.19. As shown, for the tested perturbations, the original RWTA network produces the best performances. The settings of SH and HA produce performances slightly lower than the original network. The settings of SA bring the most degradation, followed by HHSA and HH. This indicates that SA connections play the most important role in training. After them are the HH connections.

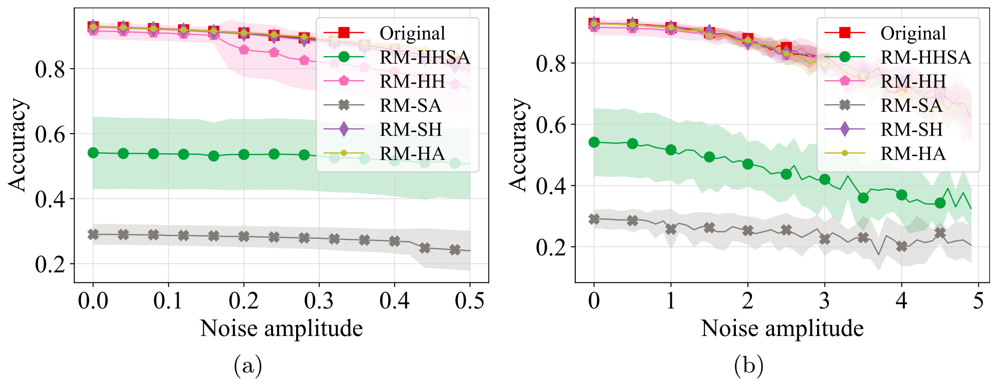


Figure 4.19: Effects of connection removal in the RWTA network in training on MNIST. (a) Input Pepper noise. (b) Network parameter uniform noise.

I emphasize that the "RM-HHSA" setting corresponds to a three-layer structure in the RWTA network that is the same as [47]. As shown, the performance of this variation is obviously worse than the original network. This indicates that under the condition of the same number of hidden neurons, extending the model from layered to fully connected improves the training performance and robustness.

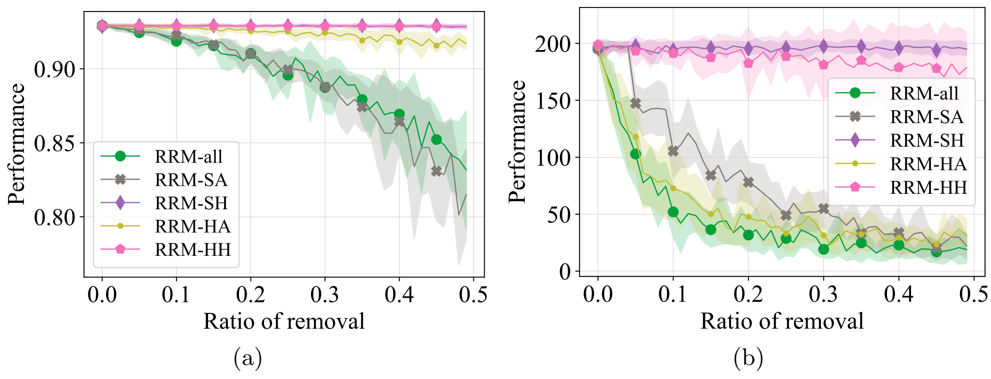


Figure 4.20: Effects of connection removal in RWTA network in testing on (a) MNIST, (b) GYMIP. "RRM" means random removal of connections.

Effect of random connection removal in testing

In this test, I look at the effects of connection removal in testing to check their contribution to testing performance. I randomly remove a number of connections from a trained RWTA network, i.e., set the weight values to 0. I perform this test on connections in different parts of the network, and use HH, HA, SA, and SH to represent these types. I use ALL to refer to removing all four types of connections at the same time. As for measuring the strength of removal, I use the ratio of the number of removed connections to the number of original connections in the corresponding type.

This test is done on MNIST and GYMIP, and the results are plotted in Figure 4.20. As shown, as the ratio of removed connections increases, the testing accuracy drops in settings ALL, HA, and SA; it does not change significantly in settings SH and HH. This indicates that the testing performance depends more on HA and SA connections than on SH and HH connections. Among HA and SA connections, the effects of removing SA connections are stronger. This indicates that SA connections play a more important role than HA in testing performance. Although in the previous test, i.e., effects of connections in training, HA connections do not contribute significantly to training, the results here show that HA connections do have effects on the testing performances. This further demonstrates the effectiveness of the fully connected network structure.

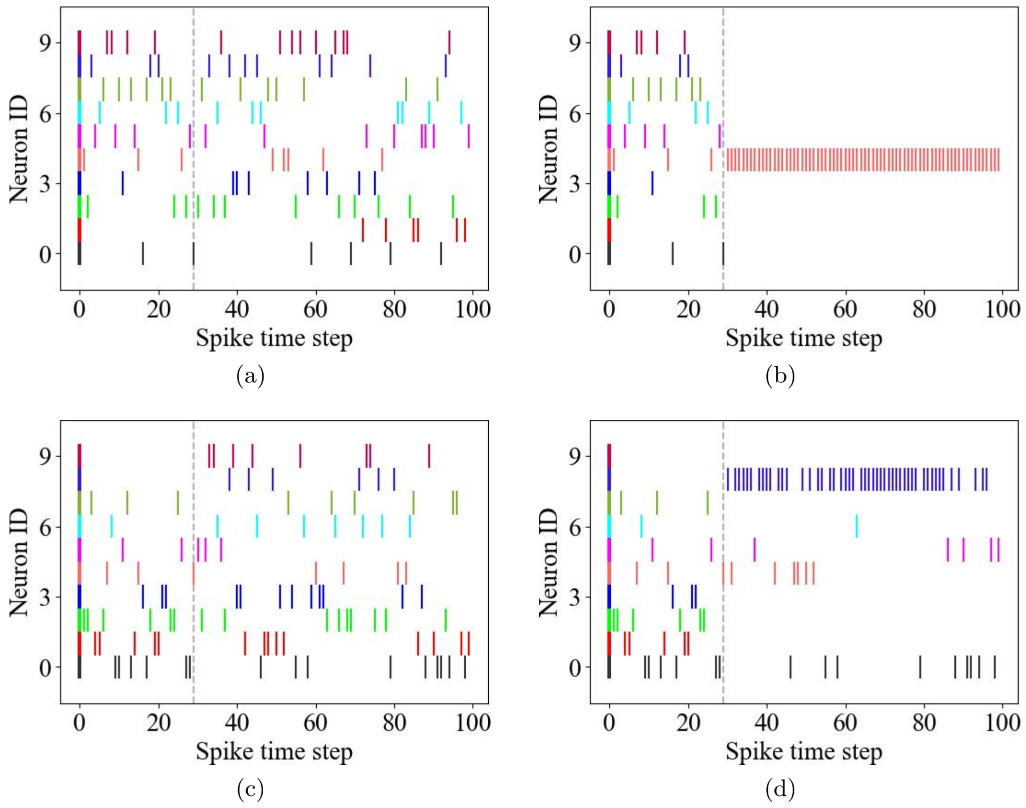


Figure 4.21: Visualization of spike trains in WTA circuits on MNIST. (a, b) Action WTA circuit. (c, d) A hidden WTA circuit. (a, c) Randomly initialized RWTA network. (b, d) Trained RWTA network.

4.3.5 Network Visualizations

Here I provide visualizations of the RWTA network for a more intuitive understanding of its properties.

Firing process visualization on MNIST

I collect the spike trains when a randomly selected image (No. 901) in the MNIST testing dataset is fed to the RWTA network, and plot them in Figure 4.21. The size of the hidden and action WTA circuits is 10. In each plot, there are 10 spike trains, each corresponding to one neuron in the WTA circuit. The lengths of the simulation time and spike response window are 100 and 30 spike time steps. In practice, I start updating the firing probabilities at the 29th spike time step, which is marked with dotted vertical lines.

As shown, in the RWTA network before training, the spike trains of the action circuit (Figure 4.21a) and hidden circuit (Figure 4.21c) are both noisy and evenly distributed across the neurons, even after some iterations of the firing probabilities. In contrast, in the trained RWTA network

(Figure 4.21b, 4.21d), the spike trains quickly gather to a few neurons during the iteration of firing probabilities. This is consistent with the previous conclusion that the inference process converges quickly on MNIST.

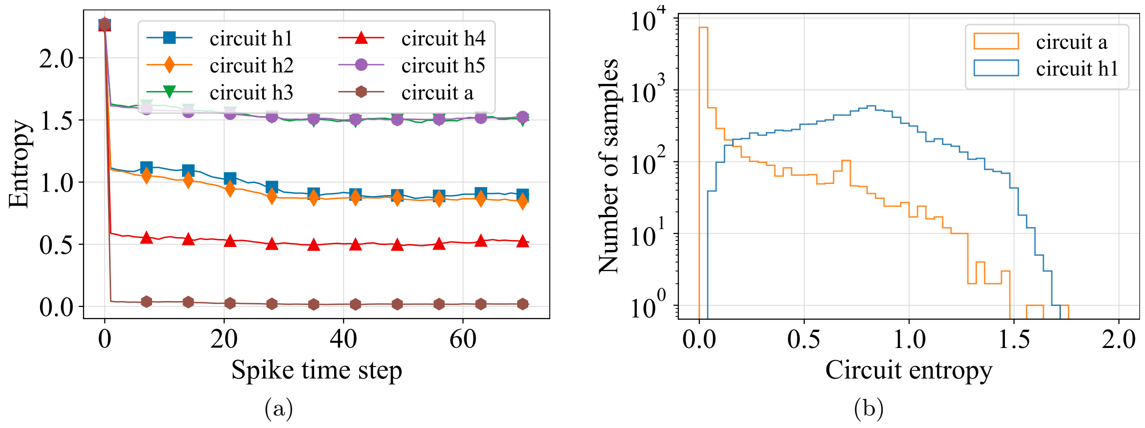


Figure 4.22: Entropy values of WTA circuits on MNIST. (a) Change of entropy values of 5 hidden circuits and the action circuit during inference. (b) Distribution of entropy values of a hidden circuit and the action circuit on MNIST.

I further calculate the entropy of the firing probability distribution in WTA circuits to measure their selectivity (after training). I first look at how the selectivity of the action WTA circuit and 5 randomly selected hidden WTA circuits changes during the inference process. In Figure 4.22a, circuits h1 to h5 are hidden circuits, and circuit a is the action circuit. As shown, in the policy inference process, the entropy of the circuits decreases sharply at the beginning and slowly afterward. The entropy of the action circuit is lower than the hidden circuits. I then look at the distribution of the entropy values when the MNIST testing set is fed to the network. As shown in Figure 4.22b, the distribution of the action WTA circuit concentrates at a value close to zero, while the distribution of the hidden circuit is smoother. These results indicate that, although the output of the RWTA network tends to converge to one neuron, the hidden part is less selective. The low selectivity in hidden circuits indicate that a source of randomness exists in the input to the action neurons in training, which may be an explanation for the robustness of the RWTA network.

Firing process visualization on GYMIP

Different from MNIST, GYMIP involves changes in the environment states, which allows visualization of how the network dynamics changes with the environment.

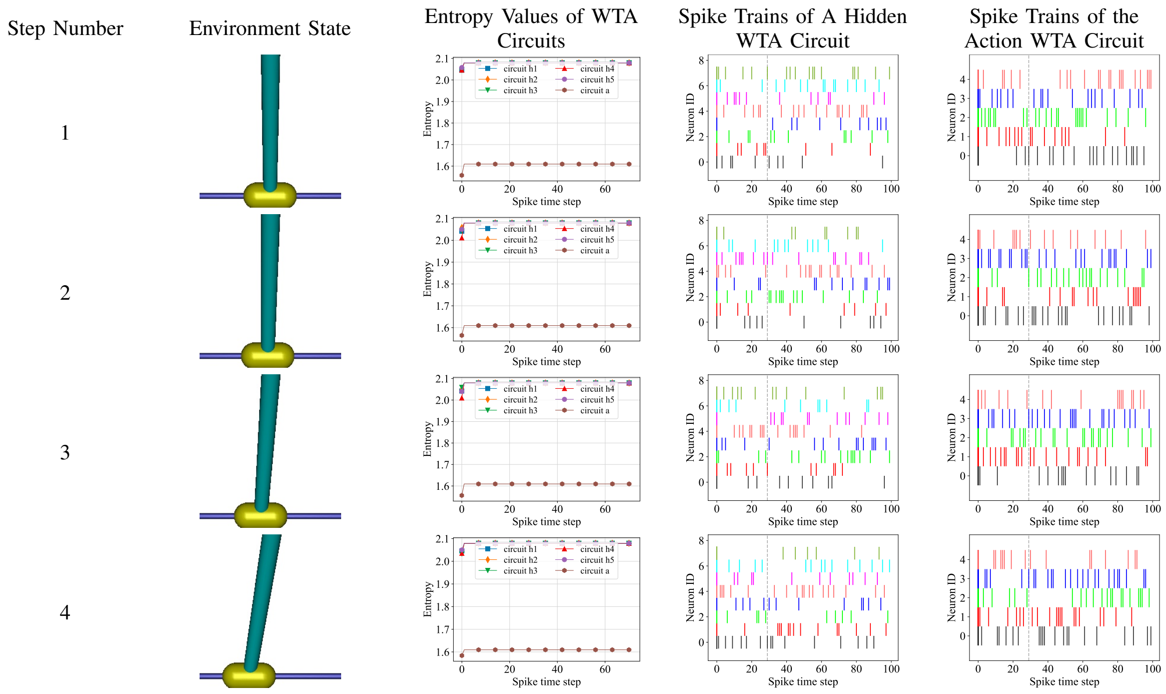


Figure 4.23: Visualization of network dynamics of SVPG on GYMIP. Before training.

I consider an untrained network and a trained network. For each network, I perform one test episode and record the network dynamics at each step. The results are presented respectively in Figure 4.23 and Figure 4.24. In these figures, the first column is the step number in the test episode. The second column is a visualization of the environment state. The third column presents the entropy values of the action WTA circuit and some hidden WTA circuits during the firing process. The last two columns respectively present the actual spike trains generated by neurons in a hidden WTA circuit and the action WTA circuit.

As shown, there is a significant difference between a random network and a trained network. In a trained network, the entropy values of WTA circuits quickly decrease in the firing process. Additionally, the firing patterns of the presented hidden and action neurons change when the environment state changes. I also notice that the action WTA circuit does not always converge to output one action quickly; instead, it may gradually change its output to the final action, as shown in step 9 and step 50 in Figure 4.24. This indicates that the firing process is necessary for the generation of the final decision output.

Distribution of parameters in RWTA network on MNIST

Here I visualize the distribution of weight values in the networks and compare SVPG with BP. To make the comparison fair, I use the SVPG-shrink variant so that the total number of

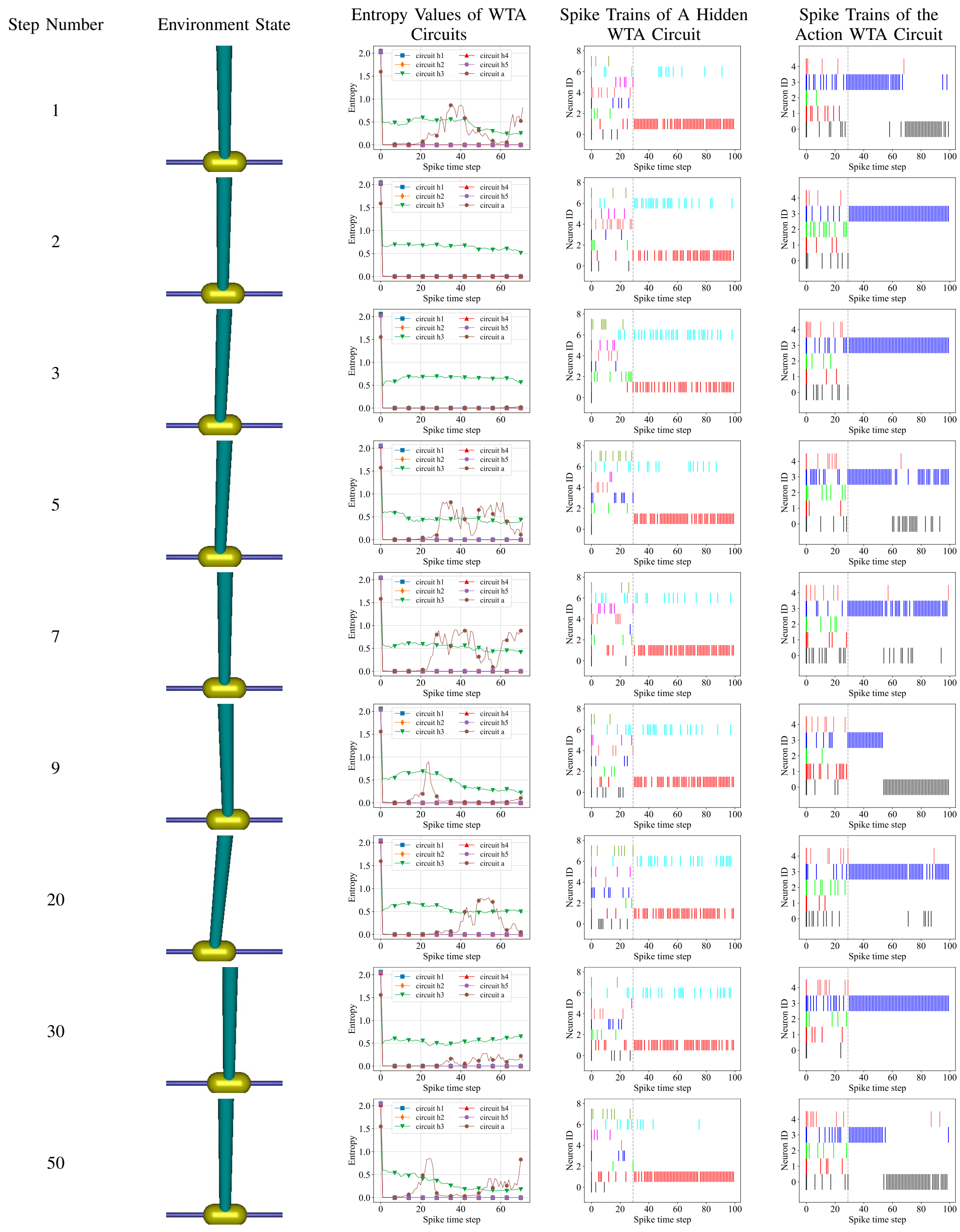


Figure 4.24: Visualization of network dynamics of SVPG on GYMIP. After training.

weight values is close to that in BP. I plot the distribution of the weight values in a trained RWTA network and an MLP in Figure 4.25a. I also plot different types of weights in the RWTA network separately in Figure 4.25b. Note that the distribution is the stack of values across 10 independent trainings.

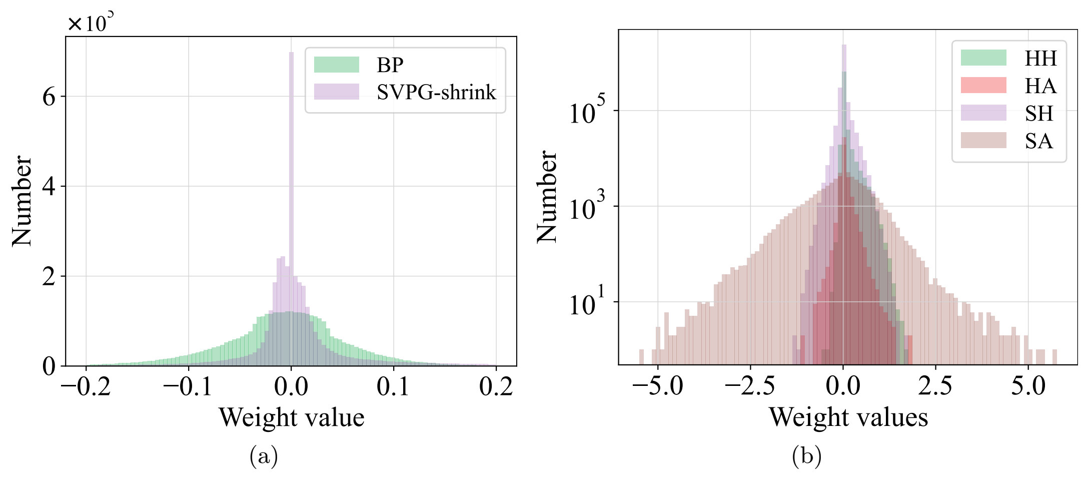


Figure 4.25: Histograms of network weight values on MNIST. (a) Comparison of SVPG-shrink and BP. (b) Comparison of different types of connections in SVPG-shrink.

As shown, the ratio of zero weights in SVPG is greater than in BP. This indicates that SVPG tends to learn a more sparse network. This can be helpful for hardware implementation because a sparse network requires less computational resource and consumes less energy. In SVPG, SH and HH types of connections exhibit a sharp spike at value 0 in their weight distributions, indicating a more distinguished sparsity in those areas when compared to the other two types. This, to some extent, explains the results in section 4.3.4, where the random removal of these connections brings less effect to the performance than other types of connections.

4.4 Chapter Summary

In this chapter, I verified the convergence assumption for the RWTA network and tested SVPG on several representative RL tasks, including a challenging DOOM task and two realistic robot control tasks. The performances demonstrated that SVPG is capable of obtaining satisfactory policies in these tasks. The robustness test revealed that SVPG can produce policies with better inherent robustness to various types of perturbations than standard methods. The test of computation costs showed that SVPG has a fast optimization speed, slow inference speed, and comparable space complexity and sample efficiency to BP. The ablation study examined the contributions of parts of the RWTA network to the performance. The visualizations revealed properties of the RWTA network including selectivity and sparsity.

Chapter 5

Reinforcement Learning for Visual Search Behaviors

Visual search is a sequential cognitive process. Understanding the mechanism behind visual search behavior is beneficial for many fields including health care. Building a brain-like model for visual search, in addition, facilitates bridging the brain structures to the functions. For extracting the motivation behind the visual search behavior, IRL methods are promising because the reward functions they learn are a natural representation of motivation. However, in existing methods, the reward functions are dependent on the agent policy [55,57–60], so they cannot reflect how the stimulus in visual search tasks affects the behavior. This chapter proposes a new IRL method, map-based maximum entropy (MME), to extract a policy-independent reward representation. Based on this, the brain-inspired SVPG method, developed in previous chapters, is applied to the visual search modeling task. MME is further applied to the pattern analysis of social and non-social visual search behaviors, which provides new insights into potential mechanisms of brain disorders.

This chapter first presents the MME method, followed by experiment results on standard visual search behaviors. This chapter then presents the results of the application of SVPG to the scanpath modeling task, as well as more in-depth analysis results on social and non-social visual search behaviors. The code for this part of the work is made publicly available at my code repository[1].

[1]`https://github.com/yzlc080733/VS_MME/`

5.1 Method Design

The design of the new IRL method aims to resolve the issue that the learned reward function is coupled with the agent policy, hindering downstream analysis of the stimulus. The new method is based on the maximum entropy IRL method, which uses a probability distribution to imitate human behavior. The method design consists of three parts. The first part is a decoupled reward function formulation. This is the core part that achieves the objective of learning a static motivation representation. A problem is that, in conventional maximum entropy IRL, the score formulation is undiscounted, which cannot preserve the sequence of human fixations. This necessitates the second design, i.e., the discounted score formulation. The third design is a constraint on the range of reward values to avoid numerical issues in practical computation.

5.1.1 Decoupled Reward Function

As reviewed in section 2.3, all IRL studies for visual search scanpath modeling that I am aware of learn a reward function that depends on state-action pairs. This is a normal choice for standard IRL tasks, but it is not ideal for the extraction of the motivation behind eye movements. The reliance of the reward function on actions makes it dynamic with respect to the search image and the target image. This makes it hard to compare the attractiveness between different regions of the search image.

From this perspective, a natural design is to learn a reward function that only depends on the state, i.e., $R(s|I_{\text{search}}, I_{\text{target}})$. This reward function can be viewed as a static map on the search image, conditioned on the target image. In the following, I use R_{map} to denote this function. However, this form of reward function is not capable of generating human-like scanpaths. A key difference between visual search and conventional IRL tasks (e.g., maze navigation) is that there is no constraint on the fixation – the agent is free to saccade from any point to an arbitrary point. Formally, this means that the state space for a certain search and target image is fully connected and that there is a one-to-one mapping $f_{a \to s}$ from the actions to the states. Therefore, the optimal action at any state is to always saccade to the point with the largest reward value: $\pi^*(s|I_{\text{search}}, I_{\text{target}}) = \arg\max_{a \in \mathcal{A}} R_{\text{map}}(f_{a \to s}(a)|I_{\text{search}}, I_{\text{target}})$. This is not similar to human scanpaths, which often contain fixations at different positions.

For this problem, I propose to add environmental rewards to the reward map. The environmental rewards are designed to capture certain properties of the scanpaths, including saccade amplitude and revisitation. The choice of saccade amplitude is inspired by previous studies like [155], which considers decreasing the likelihood of long saccades. The revisitation is inspired by the IOR mechanism, often used in the literature [50, 139]. Specifically, the saccade amplitude reward is triggered when the distance from the current fixation point f_{t-1} to the next fixation point f_t is larger than a threshold. The revisitation penalty is triggered when the next fixation point exists in the ongoing scanpath. Formally, they are defined as:

$$R_{\text{revisit}}(a_t|a_0,\dots,a_{t-1}) = \begin{cases} -\alpha_{\text{revisit}} & \text{if } \exists k \in \{0,\dots,t-1\}, \text{ s.t. } a_t = a_k, \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

$$R_{\text{amplitude}}(a_t|a_{t-1}) = \begin{cases} -1 & \text{if } \|a_t - a_{t-1}\|_2 > \alpha_{\text{amplitude}}\text{size}_{\text{search,diag}}, \\ 0 & \text{otherwise,} \end{cases} \tag{5.2}$$

where $\text{size}_{\text{search,diag}}$ is the length of the diagonal of the search image, $\alpha_{\text{amplitude}}$ and α_{revisit} are positive hyperparameters, $(a_t - a_{t-1})$ means the saccade vector between the two fixation points and the unit length is one pixel. The combined reward function is then

$$\begin{aligned} R(s_t, a_t|I_{\text{search}}, I_{\text{target}}, a_0,\dots,a_{t-1}) =& R_{\text{map}}(s_t|I_{\text{search}}, I_{\text{target}}) \\ &+ R_{\text{revisit}}(a_t|a_0,\dots,a_{t-1}) + R_{\text{amplitude}}(a_t|a_{t-1}) \ . \end{aligned} \tag{5.3}$$

With suitably set hyperparameters $\alpha_{\text{amplitude}}$ and α_{revisit}, this reward function encourages the agent to saccade away from visited points and to keep a normal amplitude of saccades. This resolves the above-mentioned problem. Note that my revisitation reward is different from IOR, which directly prohibits revisitation. Considering that human scanpaths often contain revisitations, especially when discretized into a course grid, my reward-based mechanism is more flexible as it allows revisitations.

As shown in Eq. (5.1) and Eq. (5.2), although the two additional rewards are dependent on the agent's history actions, they are independent of the search image and target image. This formulation of the reward function Eq. (5.3) can be viewed as decoupling the reward function into two parts – an image-dependent part and an agent-dependent part. In contrast, in existing IRL studies on visual search, the reward function is determined by the agent's action, which further relies on the image observations.

Since my target is an explicit reward map R_{map}, GAIL [164] and IQ-Learn [165] do not suit my needs. The maximum entropy IRL method [160] is suitable because it learns a reward function directly. Although the transition function is known in visual search, the state space is high-dimensional. Therefore, I adopt the sample-based learning method from guided cost learning [161].

5.1.2 Discounted Score

The decoupled reward function formulation facilitates learning of the static reward map while enabling the agent to perform sequences of fixations. However, standard maximum entropy IRL methods do not capture the sequence of fixations by humans. This is because of the undiscounted form of the score function. In this study, I propose to use a discounted score function, i.e., setting $\gamma < 1$. Below I give a toy example to demonstrate the effect of the discount factor in the score function.

Suppose there is only one human scanpath in the dataset, and the action space is discretized into a 1×2 grid. This means there are only two states, which I denote as s_A and s_B. The actions for fixating on these states are a_A and a_B. I omit the notation of the search and target images for conciseness. Suppose the human scanpath has $T = 2$ and $\xi_h = (s_0, a_A, s_A, a_B, s_B)$. In this example, the agent's scanpath space is $\Xi = \{(s_0, a_A, s_A, a_B, s_B), (s_0, a_B, s_B, a_A, s_A), (s_0, a_A, s_A, a_A, s_A), (s_0, a_B, s_B, a_B, s_B)\}$. Then the log-likelihood function in Eq. (2.12) is

$$\begin{aligned}\log[p(\xi_h)] =& c(\xi_h) - \log\left\{\sum_{\xi_a \in \Xi} \exp[c(\xi_a)]\right\}\\ =& R_{\text{map}}(s_A) + \gamma R_{\text{map}}(s_B)\\ &- \log\big\{\exp[R_{\text{map}}(s_A) + \gamma R_{\text{map}}(s_B)] + \exp[R_{\text{map}}(s_B) + \gamma R_{\text{map}}(s_A)]\\ &\qquad \exp[R_{\text{map}}(s_A) + \gamma R_{\text{map}}(s_A)] + \exp[R_{\text{map}}(s_B) + \gamma R_{\text{map}}(s_B)]\big\}.\end{aligned} \tag{5.4}$$

For succinctness, I use r_A to denote $R_{\text{map}}(s_A)$ and r_B to denote $R_{\text{map}}(s_B)$. The partial derivation of the target over the rewards is

$$\begin{aligned}\frac{\partial \log[p(\xi_h)]}{\partial r_A} =& \frac{(1-\gamma)e^{r_B+\gamma r_A} + (-\gamma)e^{r_A+\gamma r_A} + e^{r_B+\gamma r_B}}{e^{r_A+\gamma r_B} + e^{r_B+\gamma r_A} + e^{r_A+\gamma r_A} + e^{r_B+\gamma r_B}}\\ \frac{\partial \log[p(\xi_h)]}{\partial r_B} =& -\frac{\partial \log[p(\xi_h)]}{\partial r_A},\end{aligned} \tag{5.5}$$

where the numerator can be written as

$$(1-\gamma)e^{r_B+\gamma r_A} + (-\gamma)e^{r_A+\gamma r_A} + e^{r_B+\gamma r_B} = e^{r_B+\gamma r_A}\left[1-\gamma-\gamma e^{r_A-r_B}+e^{-\gamma(r_A-r_B)}\right], \quad (5.6)$$

which is monotonically decreasing with reference to $e^{r_A-r_B}$. (1) When $\gamma = 1$, $\frac{\partial \log[p(\xi_h)]}{\partial r_A}$ is negative when $r_A > r_B$, positive when $r_A < r_B$, and is zero when $r_A = r_B$. The opposite is true for $\frac{\partial \log[p(\xi_h)]}{\partial r_B}$. This means the IRL target tends to make the reward values closer to each other during training. (2) In contrast, when $\gamma < 1$, there is a certain threshold $r_\Delta > 0$ such that $\frac{\partial \log[p(\xi_h)]}{\partial r_A}$ is positive when $r_A - r_B < r_\Delta$. The differential is only negative when $r_A - r_B > r_\Delta$. This means the objective tends to make $r_A > r_B$. This is good for replicating the ordering of fixations in the human scanpath.

The above example presents a toy scene where the agent scanpaths can be enumerated. In practice, the agent scanpaths are sampled according to the agent policy, which may be noisy or even biased. I further examine the effect of the discounted score in the ablation study, which will be introduced in section 5.2.5.

Note that the extension of standard maximum entropy IRL to discounted score functions has been studied in a previous work [198]. A further extension proposes to estimate an appropriate discount factor [199]. My contribution is to emphasize the importance of a discounted score for map-based IRL for visual search.

5.1.3 Constraint on Value Range

The original objective function Eq. (2.12) does not guarantee a value range for the learned reward values. Previous studies add regularization terms, e.g., weight decay [200] and encouragement for monotonic decrease in values [161], to the loss function. These methods can, to some extent, prevent the reward values from growing to infinity. However, there is no clear upper bound for the values.

I observe that in the calculation of the probability value Eq. (2.11), the exponential of the score can be intractably large when the rewards are too large. Specifically, when the discounted sum of rewards along a sampled or human scanpath exceeds 89, the exponential value will be greater than 4.49×10^{38}, exceeding the largest representable number of "float32", the default data type in PyTorch [182]. This motivates me to set up a fixed bound for the sum of all reward values.

To facilitate the comparison of rewards between different images, I also set a fixed value range for each reward value. These rules are designed as regularization terms for the loss function, as shown in Eq. (5.7) and Eq. (5.8) below.

$$\begin{aligned}\mathcal{L}_{\text{reg,upper}}(I_{\text{search}}, I_{\text{target}}) =& \frac{1}{2}\sum_{s_t}[R_{\text{map}}\,(s_t|\cdot)^2] \\ & \cdot \max\left\{f_{\text{ReLU}}[\max_{s_t} R_{\text{map}}(s_t|\cdot) - \beta_{\text{max}}], f_{\text{ReLU}}[\sum_{s_t} R_{\text{map}}(s_t|\cdot) - \beta_{\text{sum}}]\right\},\end{aligned} \tag{5.7}$$

$$\mathcal{L}_{\text{reg,lower}}(I_{\text{search}}, I_{\text{target}}) = \sum_{s_t} f_{\text{ReLU}}[-R_{\text{map}}(s_t|\cdot)], \tag{5.8}$$

where $f_{\text{ReLU}}(x) = \max(x, 0)$. The $I_{\text{search}}, I_{\text{target}}$ in the notation of R_{map} is omitted for conciseness. For the upper bound, all the values in the reward map are reduced together when the sum of the values or the maximum value exceeds the corresponding threshold β_{sum}, β_{max}. By using the squared loss, the amount of reduction is proportional to the original values. This 1) keeps the original order of the reward values, and 2) prevents the reward values from being negative. For the lower bound, each reward value is tuned separately when it falls below zero.

With the above two regularization terms, the overall objective function is

$$\begin{aligned}\mathcal{L}(\theta_r) =& \frac{1}{|\mathcal{D}|}\sum_{(\xi_h, I_{\text{search}}, I_{\text{target}})\in\mathcal{D}} \Big\{ \log[p(\xi_h|\theta_r)] \\ & - \alpha_{\text{upper}}\mathcal{L}_{\text{reg,upper}}(I_{\text{search}}, I_{\text{target}}) - \alpha_{\text{lower}}\mathcal{L}_{\text{reg,lower}}(I_{\text{search}}, I_{\text{target}})\Big\},\end{aligned} \tag{5.9}$$

where α_{upper} and α_{lower} are hyperparameters for weighting the losses.

5.1.4 Implementation Details

To implement the above designs, I create an algorithm based on the guided cost learning method [161]. Here I introduce the details of the algorithm design, including state processing, environment model, and sample augmentation.

State processing. It is widely believed [55, 57, 60, 145] that human eyes perceive visual information in a different way from cameras. The information near the fixation point is perceived with a higher resolution, while other parts are perceived with a lower resolution. Therefore, following these existing works, I also consider two views of the search image – a low-resolution full view and a high-resolution partial view at the fixation point.

Specifically, given a search image, I first resize it to the shape of 1024×768. In the following text, unless otherwise specified, the resizing of images is performed with bilinear interpolation. The low-resolution full view is constructed by resizing the search image to the size of 256×192. For the partial observation, I design it to center at the fixation point, and the field of view has a size of 40% of the search image. The observed part of the image is resized to the shape of 256×192, so it has a higher resolution than the processed search image. Note that the fixations can be near the borders of the search image, which means the field of view contains areas outside the search image. In my implementation, the outliers are filled with a gray background.

The target images are cropped from the search image. In practice, the image patches can have different sizes. To make the size of the target images consistent, I fit them to the size of 128×96. The image patch is resized to be as large as possible but within the limit of 128×96 and put at the center of the target image. The background is also set to be gray. Note that the resizing of the image patch takes into consideration the ratio of the length and width so that the processed search image and target image have the same ratio for their contents.

Note that previous visual search studies often use a feature extractor in processing the search image. For example, in [55], a Panoptic-FPN network is used to extract segmentation maps for different categories of objects in the image. In [60], a fixation density map is predicted to be observed by the IRL agent. These representations replace the original textures with semantic-based or saliency-related information, thus helping to improve the generalizability of the models. In this thesis, the model is applied to different types of visual search datasets, including both natural and synthetic ones. A single feature extractor may not work well on all the datasets. Thus, I choose to build my model on the raw images. Future work may consider extensions with different feature extractors.

Figure 5.1 shows a pair of search image and target image. They are drawn for illustration only and are not from the datasets I use.

Environment model. I design the agent to start from the center of the search image. This follows the design in [155, 201]. Note that this starting point does not count toward the length of the agent scanpath. At each fixation step, the agent obtains a high-resolution partial observation of the search image. To incorporate historical observations, I keep a record of two recent observations. For the first step, the low-resolution full view is used to pad the history

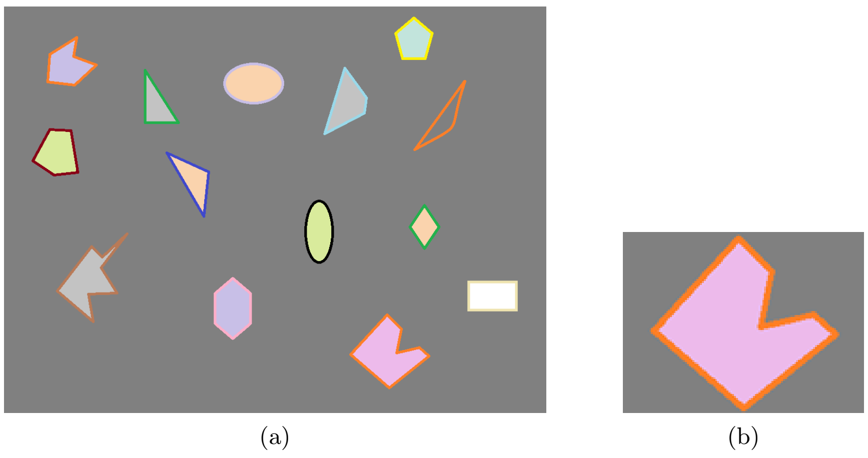


Figure 5.1: Examples of pre-processed (a) search image and (b) target image.

record. Apart from this history record, the agent also has access to the target image, which remains unchanged during the visual search process. Figure 5.2 shows the agent's perceptions in an example scanpath.

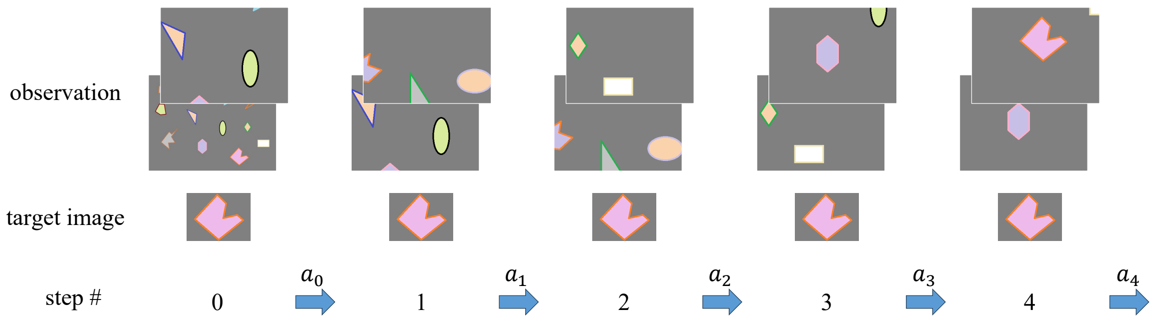


Figure 5.2: Illustration of state observations along a scanpath.

For the action discretization, I set the grid size $\text{size}_{\text{grid},x} = 8$ and $\text{size}_{\text{grid},y} = 6$. This results in 48 actions in total.

For the termination condition, I set that the agent always performs the same number of fixations as humans. This enables the application of fixation-pair-based similarity measurements. To implement this setting, the agent's scanpath sampling is conditioned on the human scanpaths.

State encoding for SVPG. The application of the SVPG method requires that the state observation be represented as a vector of firing probabilities. In this visual search modeling task, I reshape the partial observation history and the target image into vectors and convert the RGB values to $[0, 1]$ by dividing by 255. Considering that a state vector that is too long induces a high memory cost for the RWTA network, I reduce the size of the partial observations and the target image. Specifically, each observation in the history record is resized from 256×192

to 80×60. The target image is resized from 128×96 to 40×30. The final state vector is a concatenation of these images and has a length of 32400. Note that this state vector is much longer than the ones in previous standard RL tasks, e.g., 4800 in the DOOM task, thus posing a greater challenge. Figure 5.3 illustrates the state processing for SVPG.

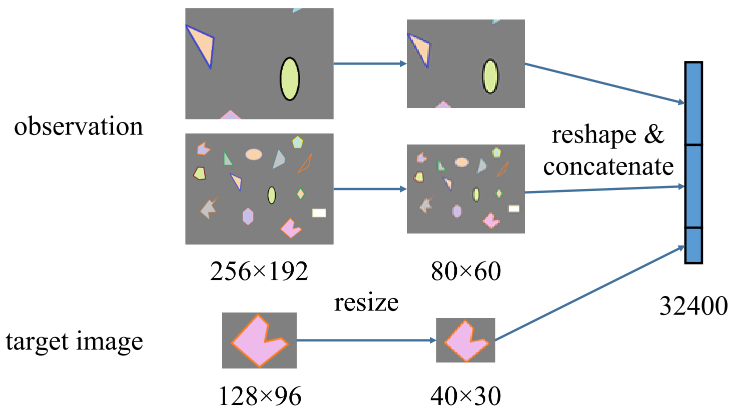


Figure 5.3: Illustration of processed state vector for SVPG.

Modifications to guided cost learning As mentioned earlier, my idea to learn an explicit reward map motivates me to choose maximum entropy IRL. Further, the high dimensionality of the state space necessitates an efficient sample-based method, so I choose to base my method on the guided cost learning [161] method.

The main features of the guided cost learning method include [161]:

1. A high-dimensional function approximator for the reward function;
2. Sample-based estimation of the partition function Z;
3. Policy optimized with entropy added to the loss function;
4. Importance sampling;
5. Sample augmentation: append human scanpaths to the agent's samples;
6. Regularization for reducing high-frequency variations and encouraging an increase in the reward values along each scanpath.

Here I keep the first three designs to facilitate learning the reward map from samples. For the importance sampling part, the original implementation keeps a record of all the scanpath samples from the agent, so the agent dataset $\mathcal{D}_a$ is a mixture of policy distributions. In my task, the high dimensionality of the agent observations and the conditions of the search image and target image make it intractable to keep a full record of agent samples. Therefore, I propose to only use the most recently sampled batch of scanpaths to update the reward model. The importance sampling weights are calculated with reference to the agent's current policy. This corresponds to the term $p(\xi_a)$ in Eq. (2.13).

About the sample augmentation technique. The original motivation is to help prevent the learning object from growing to infinity in practice [161]. In my case, there are already regularization terms in the objective function Eq. (5.9) to constrain the reward values. Therefore, I remove this sample augmentation technique. An ablation study is done to check the effect of sample augmentation. As will be shown in section 5.2.5, the augmentation of human scanpaths to the agent samples makes the training progress slower.

The regularizations on reward values are removed in my method. In my task, with the effect of the revisitation penalty, an ideal form of reward value should be decreasing along the scanpath, which is contrary to the original design [161]. The other regularization, which aims to reduce high-frequency variations, could be helpful for learning a smoother reward map and improving generalizability. Future work may investigate its effect on learning performance.

Apart from these modifications, I also propose some other designs to the IRL method. (1) Because the reward model is updated in an online manner, I increase the randomness of the scanpaths sampled by the agent. This is implemented by smoothing the policy distribution. Specifically, the policy π' used for sampling scanpaths for the reward model is obtained by adding a constant value to the original policy π and re-normalizing:

$$\pi'(s_t, a_t) = \frac{\pi(s_t, a_t) + \nu}{\sum_{a \in \mathcal{A}}[\pi(s_t, a) + \nu]}, \tag{5.10}$$

where ν is the constant. In my experiments, ν is set to 0.1.

(2) For the training of the agent, I use the PPO algorithm [65]. Because the reward function is changing in training, I add a greedy policy to the sampling process to accelerate the learning of the agent. The greedy policy selects actions according to the reward map predicted by the reward model. To simulate the revisitation penalty, the penalty value α_{revisit} is subtracted from the reward map after each visitation. During training, I use the greedy policy π_{greedy} randomly with a probability of 0.3, and use the original agent policy for the rest of the time.

Function approximators. I propose two implementations of my method. One is based on a conventional ANN, and the other is the RWTA network trained by SVPG.

The ANN-based implementation is for verifying the effectiveness of my IRL algorithm. It offers a better training speed and lower memory cost. In my implementation, the ANN consists of two convolutional blocks and some fully connected layers. One convolutional block processes the observation history and the other processes the target image. The output features of these two blocks are concatenated and then input to the fully connected layers. The output of the fully connected layers is normalized with the softmax function and its size is 48, corresponding to the possible fixation positions.

The SVPG-based implementation is for testing the effectiveness of SVPG in solving the visual search IRL task. The RWTA network is configured to have 32400 state neurons, 100 hidden WTA circuits, each with 10 neurons, and 48 action neurons corresponding to the possible fixation positions.

For the above two implementations, a critic model is also built to learn the Q-values for the agent, which is part of the PPO algorithm [65]. The critic model is implemented with a network whose shape is mostly the same as the ANN-based agent, except that the final softmax function is removed.

5.1.5 MME Algorithm

Since my method belongs to the maximum entropy IRL methods and features learning a reward map, I name it as map-based maximal entropy (MME) IRL, and use MME in the following text for succinctness.

The MME algorithm is summarized in Algorithm 3. In my experiments, the default values for some of the hyperparameters are that $\gamma_r = 0.9$, $\alpha_{\text{revisit}} = 10$, $\alpha_{\text{amplitude}} = 1$ (indicating absence of saccade amplitude penalty), $\beta_{\text{max}} = 10$, $\beta_{\text{sum}} = 40$, $\alpha_{\text{upper}} = 1$, and $\alpha_{\text{lower}} = 5$. MME works in epochs and each epoch contains two parts. Firstly, a number of human scanpaths are sampled from the dataset, episodes are run to collect agent scanpaths, and the agent is updated using current estimation of the reward map. Then, some more episodes are run to collect agent samples, evaluate human and agent scanpaths, and update the reward model.

Algorithm 3 map-based maximal entropy IRL (MME)

Input: Set of human scanpath $\mathcal{D}$,
Parameter: γ_r, $\text{size}_{\text{grid},x}$, $\text{size}_{\text{grid},y}$, α_{revisit}, $\alpha_{\text{amplitude}}$, β_{max}, β_{sum}, α_{upper}, α_{lower}, N_{train}, N_{agent}, N_{reward}, N_{sample},
Output: Reward model parameter θ_r, agent model parameter θ_a.

```
1: Initialize $\theta_r$, $\theta_a$.
2: for Epoch = 1, ..., $N_{\text{train}}$ do
3:   for Agent episode = 1, ..., $N_{\text{agent}}$ do {Train agent}
4:     Sample $(\xi, I_{\text{search}}, I_{\text{target}})$ from $\mathcal{D}$.
5:     Calculate $R_{\text{map}}(s_t|I_{\text{search}}, I_{\text{target}})$ with $\theta_r$.
6:     With 0.3 probability, obtain scanpath $\xi_a$ with $\pi_{\text{greedy}}$;
7:     Otherwise, predict scanpath $\xi_a$ with $\pi$.
8:     Evaluate $\xi_a$ with $R$ according to Eq. (5.3).
9:     Update agent model $\theta_a$ with the PPO algorithm.
10:  end for
11:  for Reward episode = 1, ..., $N_{\text{reward}}$ do {Train reward}
12:    Sample $(\xi, I_{\text{search}}, I_{\text{target}})$ from $\mathcal{D}$.
13:    Calculate $R_{\text{map}}(s_t|I_{\text{search}}, I_{\text{target}})$ with $\theta_r$.
14:    Evaluate $\xi$ with $R$ according to Eq. (5.3).
15:    Calculate score $c(\xi)$ according to Eq. (2.7), with reward gamma $\gamma_r$.
16:    for Agent sample $i$ = 1, ..., $N_{\text{sample}}$ do
17:      Predict scanpath $\xi_{a,i}$ with smoothed policy $\pi'$.
18:      Calculate score $c(\xi_{a,i})$ according to Eq. (2.7), with reward gamma $\gamma_r$.
19:    end for
20:    Update $\theta_r$ with $\mathcal{L}(\theta_r)$ according to Eq. (5.9) and Eq. (2.13).
21:  end for
22: end for
```

5.2 Experiments on Visual Search

I apply the MME method to different visual search datasets and compare it with standard visual search methods to verify its effectiveness. The experiments contain four parts.

1. To verify the ability of MME in replicating human scanpaths, I test the ANN-based implementation on three datasets and compare it with standard methods.

2. I look at the effects of different settings of the environmental reward functions on the training performance.

3. I perform ablation studies to check the contribution of designs in MME to the performance.

4. I apply SVPG to the training of the agent to check the performance of SVPG in solving this IRL task.

5.2.1 Datasets and Measurements

I consider three visual search datasets, namely COCO-Search18 [140], IVSN [50], and ASD [30]. The COCO-Search18 dataset features natural search images and is popular among recent visual search studies. The IVSN dataset is picked from the object array experiment in reference [50]. The search images are synthetic, containing 6 small images. Both the COCO-Search18 and the IVSN datasets are collected from normal participants. The ASD dataset includes scanpaths from different types of subjects, including people with amygdala lesions and people with ASD. The search images in the ASD dataset are synthetic and contain 24 objects (presented as small image patches) whose social/non-social category information is available. The target images are also classified as social or non-social. Therefore, the ASD dataset facilitates investigation into the effect of social/non-social items and the behavior patterns of different types of subjects.

Dataset pre-processing. I first discard target-absent scanpaths and only keep the ones in which the target image exists in the search image. To use the MultiMatch similarity metric, I remove scanpaths that contain only one fixation, so that all scanpaths I use have a minimum length of three when the initial fixation is prepended. I also notice that some human scanpaths contain fixations outside the search images. For these outlier points, I constrain their coordinates to the size of the search image.

For the IVSN dataset, the available eye movement data is raw and not sorted into fixations. I generate fixations from the raw coordinate data by clustering points that are close to each other. The threshold for the clustering is set to 5.

The ASD dataset contains scanpaths from 50 subjects, among which 13 subjects have epilepsy, 3 subjects have amygdala lesions, 2 subjects are labeled as "stroke", 13 subjects have ASD, 8 subjects form the control group of ASD, and 11 subjects are from the NUS as another control group [30]. After screening, each subject has around 80 scanpaths for social-item-targeted scanpaths and around 80 for non-social-item-targeted scanpaths. The total number of scanpaths in the ASD dataset is 7413.

To save the computational cost, I shrink the COCO-Search18 and IVSN datasets. This is done by randomly sampling 80 scanpaths for each subject. There are respectively 10 and 16 subjects in the COCO-Search18 and the IVSN dataset. Therefore, after sampling, there are 800 and 1280 scanpaths in total. The scanpaths for each subject, having a number similar to the ASD dataset, enable comparison of models trained on subject-wise sub-datasets. Abusing notation, in the following, I still use “COCO-Search18” and “IVSN” to refer to these shrunk datasets. The performance of methods on the full set of COCO-Search18 and IVSN datasets is left for future work.

Table 5.1: Scanpath length statistics.

Dataset	Min length	Max length	Mean Length
COCO-Search18	2	34	3.875
IVSN	2	54	7.073
ASD	2	78	7.660

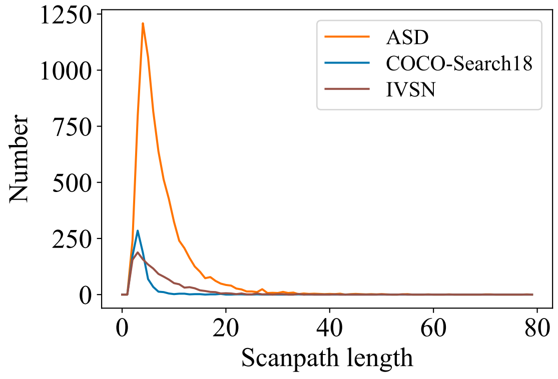


Figure 5.4: Histograms of scanpath lengths on the three datasets.

Table 5.1 and Figure 5.4 show the statistics and histograms of the scanpath lengths on the three datasets. As shown, most scanpaths have a length shorter than 10.

Measurements. For the measurements, I use the Levenshtein similarity [169], MultiMatch [168], and average point distance. The Levenshtein similarity neglects the spatial relationship between fixations, while the MultiMatch metric keeps it. The average point distance is similar to the position dimension of the MultiMatch metric, but it does not simplify the scanpaths. Therefore, it is more sensitive to subtle differences in fixation positions.

In practice, the Levenshtein similarity requires a grid size to convert pixel-valued coordinates to letters. I choose the grid size to be 8×6. The implementation is based on a public code repository[2]. Specifically, each grid is assigned a different letter. This transforms the two scanpaths to be compared into two strings. The minimum number of edits is calculated as the Levenshtein distance value. The allowed types of edits include replacement, insertion, and deletion. The

[2]`https://github.com/rapidfuzz/Levenshtein/`

Levenshtein distance can be calculated in a recursive way as shown below:

$$d\left(h_1, h_2\right) = \begin{cases} |h_1| & \text{if } |h_2| = 0, \\ |h_2| & \text{if } |h_1| = 0, \\ d\left(\mathrm{rm}(h_1), \mathrm{rm}(h_2)\right) & \text{if } \mathrm{first}(h_1) = \mathrm{first}(h_2), \\ 1 + \min\Big\{ d\left(\mathrm{rm}(h_1), \mathrm{rm}(h_2)\right), d\left(h_1, \mathrm{rm}(h_2)\right), & \\ \qquad d\left(\mathrm{rm}(h_1), h_2\right) \Big\} & \text{otherwise.} \end{cases} \tag{5.11}$$

where h_1 and h_2 are two strings to be compared, $\mathrm{first}(\cdot)$ is the first letter in a string, $\mathrm{rm}(\cdot)$ is the string with the first letter removed, $|\cdot|$ is the length of a string, and $d\left(h_1, h_2\right)$ is the Levenshtein distance between string h_1 and h_2. Given the Levenshtein distance, the Levenshtein similarity value $m_{\text{Levenshtein}}$ is calculated as

$$m_{\text{Levenshtein}} = 1 - \frac{d\left(h_1, h_2\right)}{|h_1| + |h_2|}. \tag{5.12}$$

The MultiMatch metric considers scanpaths as sequences of two-dimensional vectors. A simplification step and an alignment of the scanpaths are performed to remove noise and extract meaningful fixations. Specifically, the simplification step combines consecutive saccades into one saccade when their amplitudes are smaller than a threshold and when their differences in angle are smaller than a threshold. The alignment operation is performed on the simplified scanpaths; a comparison matrix is constructed where each element is the vector difference between each pair of saccades; the Dijkstra algorithm is used to find the shortest path that goes from the top left element to the bottom right element, with the matrix values regarded as cost values; only movements that goes down, right, or lower right are allowed to preserve the sequence of fixations. After these steps, the two scanpaths to be compared have the same length. MultiMatch reports five similarity values, respectively characterizing the average differences in shape (i.e., saccade vector difference), length, direction (i.e., angular difference), fixation position, and fixation duration. To normalize the values to the range of $[0, 1]$, the first three values are divided by the diagonal of the search image, the direction value is divided by π, and the duration value is divided by the maximum duration value being compared. In this thesis, the duration similarity value is discarded because fixation duration is not modeled. The implementation is based on a public code repository[3].

The average point distance is calculated as the mean Euclidean distance between fixation points at the same step in a scanpath and is noted as "point distance" in the following text. Specifically, the distance value $m_{\text{point distance}}$ is defined as

$$m_{\text{point distance}}(\xi_a, \xi_h) = \frac{1}{T} \sum_{t=0}^{T-1} \left[(x_{a,t} - x_{h,t})^2 + (y_{a,t} - y_{h,t})^2 \right]^{\frac{1}{2}}, \tag{5.13}$$

where the scanpaths ξ_a and ξ_h are assumed to have the same length T, and the subscripts a and h respectively means the coordinate values of the fixations of the agent and human.

5.2.2 Methods for Comparison

For the comparison methods, I choose representatives for different types of predictive visual search models. The categories include unsupervised, supervised, RL, and IRL.

IVSN. This is selected as a representative of unsupervised saliency prediction methods. The implementation is based on [142]. To apply the feature extractor, the search image is cropped into patches of size 224 × 224. The target image is resized to 32 × 32. A VGG16 network pre-trained on the ImageNet dataset is used to extract the features and calculate the saliency map. An IOR mechanism is used to generate the scanpath from the saliency map. At each step, the point with the largest saliency value is selected as the next fixation, and then the saliency values in a square area around the point are set to zero, implementing the inhibition. The side length of the square is set to 210 for the ASD dataset, 100 for the COCO-Search18 dataset, and 170 for the IVSN dataset.

nnIBS. This is selected as a representative of unsupervised scanpath prediction methods. The method is from references [142, 157] and the implementation is based on the code for [142]. The prior distribution is calculated with DeepGazeIIE, and the target similarity map is calculated using the IVSN method. The position of fixation is discretized into a grid where each cell is square and the side length is 32 pixels. The other hyperparameters are kept the same as in the implementation in [142].

[3]`https://github.com/adswa/multimatch_gaze`

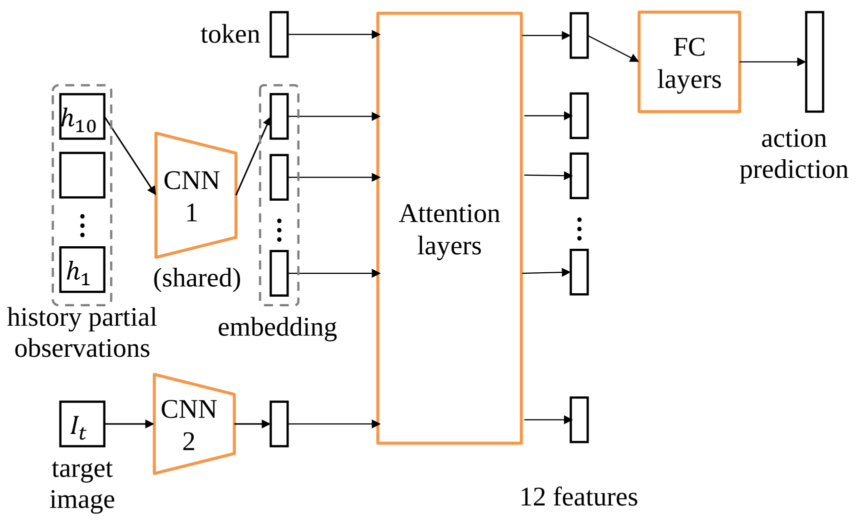


Figure 5.5: Network structure for attention analysis.

Behavior cloning. This is representative of supervised learning methods. I use a Transformer as the function approximator. The model takes the most recent 10 partial observations and the target image as input and outputs the distribution of the next fixation. Figure 5.5 presents the structure of the Transformer. The token is sampled from a normal distribution. For early fixations, the history is padded with gray images. The action space is evenly discretized into an 8×6 grid. The model is trained with the cross-entropy loss. In the following text, I use "BCtransformer" to denote this method.

RL. This is selected to represent RL methods. The method is from reference [54]. The reward generation is based on the ScanMatch similarity metric. The search image is input to the model. The attention map required by the model is generated using the ground truth, i.e., a binary map where the area for the target image is marked with 1 and elsewhere 0. The action space is discretized using a 40×30 grid. Hyperparameters for training are kept the same as in [54].

GAIL. This is selected to represent IRL methods. The implementation is based on [142] and [55]. The code from [142] is used to build the high-resolution and low-resolution belief maps. Then, the code from [55] is used to train the agent and the discriminator. To enable the use of the original code, the search image is resized to 520×312, and the target image has a shape of 128×96. The action space is discretized to 32×20. Hyperparameters for training are kept the same as in [55].

		Entropy				
Reward Learning Rate	Agent Learning Rate	0.5	1	2	4	8
0.00001	0.00001	268.338	266.571	265.741	266.404	267.714
	0.0001	268.016	267.12	264.957	269.317	266.712
0.0001	0.00001	263.269	263.859	**258.897**	259.131	260.755
	0.0001	270.674	267.315	263.617	263.559	266.412

Figure 5.6: Hyperparameter tuning of MME on the ASD dataset.

Random uniform distribution. This is added as a baseline model. The agent always samples an action from a uniform distribution over the search image. The action is represented in pixels, i.e., without action discretization. In the following, I use "randuniform" to refer to this model.

For all the above methods, the termination is set to make the length of the agent's scanpath the same as humans. For methods that require resizing the search image, the fixation coordinate values of human scanpaths are transformed accordingly. After training, the predicted fixations are converted back to the original size of the search image to ensure the fairness of the comparison.

5.2.3 Benchmark

I compare the performance of MME to other standard visual search predictive methods. This is based on the ANN-implementation of MME. In this experiment, the datasets are randomly split into a training set and a validation set, each containing 90% and 10% of the scanpaths.

I begin with the hyperparameter tuning for MME. This is done on the ASD dataset. The learning rates of the agent and reward model and the entropy ratio are tuned. Each setting is run once. The models are trained for 30k steps, and a checkpoint is saved every 2k steps. All the checkpoints are evaluated on the validation set, and the best performance is reported. Note that the agent acts greedily in the validation, i.e., always selects the action with the largest probability. To compare the performance of different hyperparameter settings, I use the point distance metric.

Figure 5.6 presents the performance values under all the settings. The best performance is marked with bold text. As shown, among the considered range of hyperparameters, the best setting is 2.0 for the entropy ratio, $1e^{-4}$ for the reward learning rate, and $1e^{-5}$ for the agent learning rate. This setting is used in the rest of the experiments unless otherwise specified.

For the BCtransformer, RL, and GAIL methods, considering that the learning target and the validation metric can be different, I save 10 checkpoints through the training process, evaluate them on all the metrics, and report the best performance. This experiment is run for 3 independent repetitions.

The results are plotted in Figure 5.7. The COCO-Search18 dataset is denoted as “COCO” for clarity. The mean performances on MultiMatch (four dimensions), Levenshtein similarity, and point distance are drawn as horizontal bars. The error bars show the standard deviations. For point distance, a smaller value means better similarity; for the other metrics, a larger value means better similarity.

From these results, it can be inferred that:

- For most methods and metrics, the performances on the COCO-Search18 dataset are higher than the ASD and the IVSN datasets. Considering that not all methods rely on pre-trained feature extractors, this means that the scanpaths in the COCO-Search18 dataset are relatively simpler to replicate. There is no significant difference between the ASD and the IVSN dataset.
- For most datasets and metrics, the random uniform distribution generates the worst performance. This indicates that all other methods, no matter pre-trained or trained, can generate scanpaths more similar to humans than random.
- In general, the unsupervised methods (i.e., IVSN and nnIBS) and GAIL have performances lower than other learning methods. For the unsupervised methods, the discrepancy between the pre-training datasets and the target dataset can be a contributor. For the GAIL method, the sensitivity to hyperparameters and training instability may contribute to the performance. Future work can consider tuning the hyperparameters instead of using the default ones.
- The best performances under different metrics are obtained by RL, BCtransformer, MME, and nnIBS. Considering the 18 combinations of dataset and metric, MME obtains the best performance in 9 cases.

I would like to emphasize that my method MME is tuned on the ASD dataset, while the other methods, although with multiple checkpoints preserved, are based on default hyperparameters. Therefore, obtaining the best performance on many metrics and datasets does not necessarily

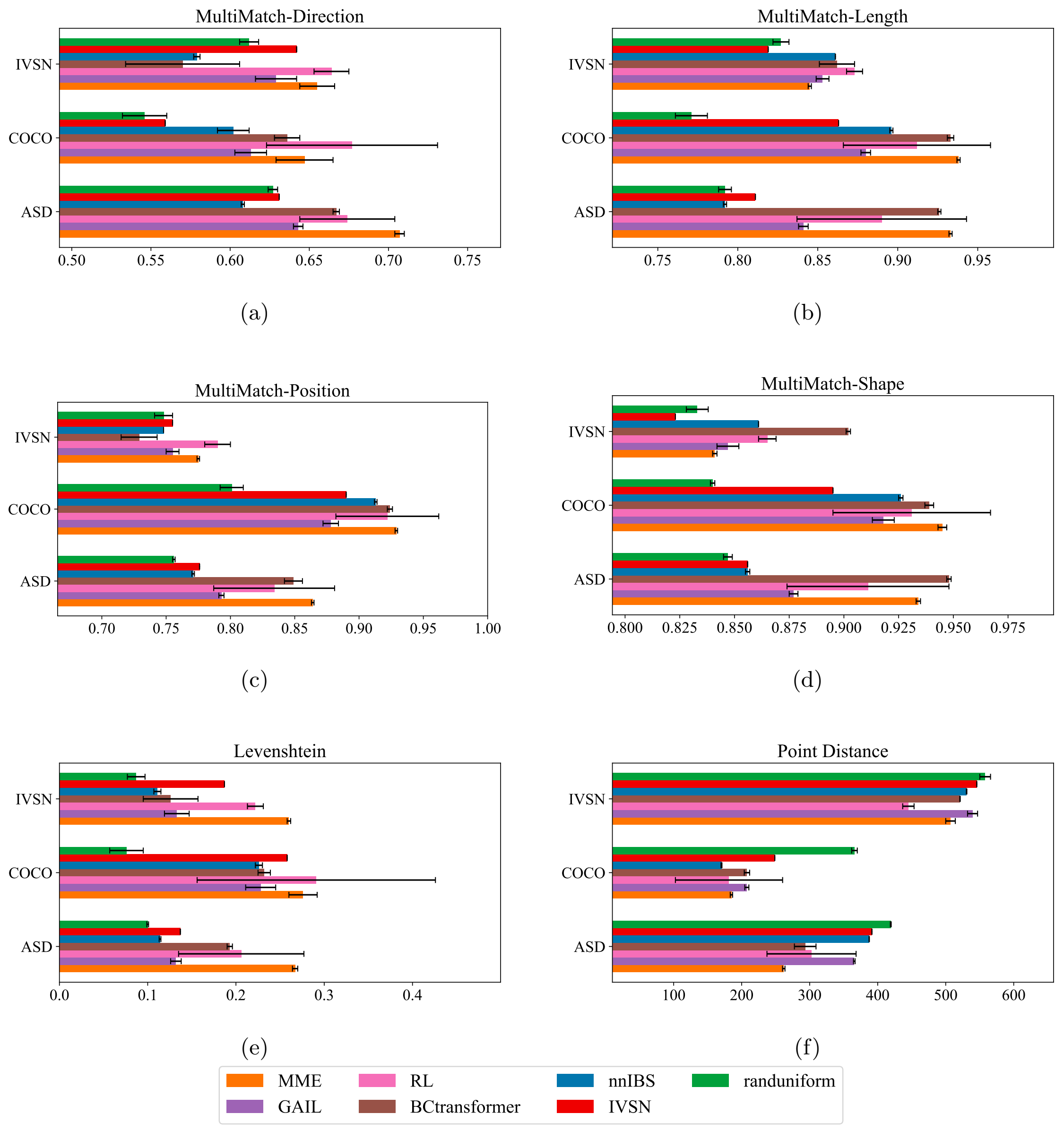


Figure 5.7: Validation performances of different methods on all the datasets.

mean MME is better than the other methods. However, the results do show that MME can get a good performance comparable to the standard visual search models.

As discussed in section 5.1.1, learning a reward map alone does not replicate human scanpaths. Here, the performance of MME indicates that the two environmental rewards are sufficient for modeling human scanpaths. In addition, the results also suggest a property of human scanpaths that they are driven by two separate forces – one based on the image stimulus and the other based on history fixations. Based on this understanding, MME makes it possible to separately analyze the effects of image stimulus and behavior patterns (specifically, revisitation and saccade amplitude). This can improve the analysis of both aspects by removing the influence of the

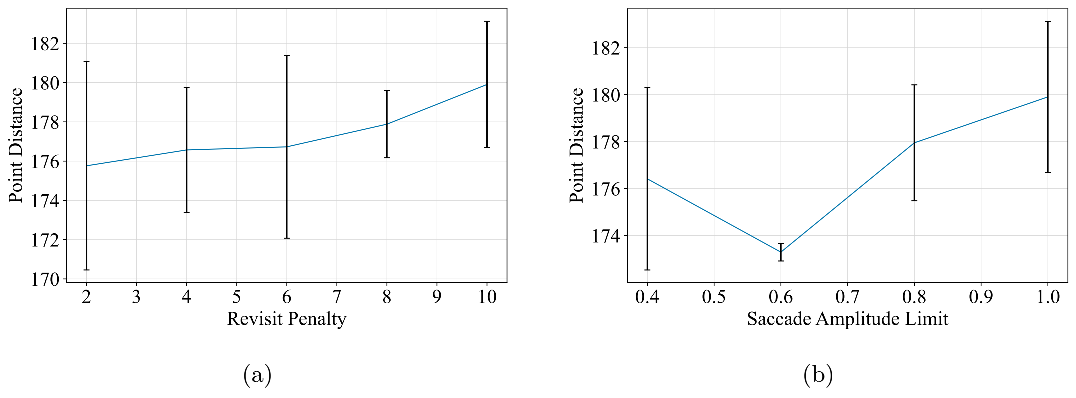


Figure 5.8: Effects of different reward settings on the performance, measured by the point distance metric.

other aspect. Potential applications include document design where the effects of document components on the observers need to be measured, as well as mental health-related research where the behavior patterns of people need to be extracted. The following section 5.2.4 provides an example analysis of the effect of environmental rewards. In section 5.3, MME is applied to the study of ASD-related analysis.

5.2.4 Effect of Environmental Rewards

In this part of the experiment, I focus on the effects of different settings of the environmental reward function on the performance. Specifically, there are two key hyperparameters, α_{revisit} and $\alpha_{\text{amplitude}}$. For α_{revisit}, a value of 0 means no penalty for revisiting fixations. Consider that I limit the maximum of the reward values to 10, a value greater than or equal to 10 will result in a negative reward for all revisitations. For $\alpha_{\text{amplitude}}$, a value of 1 means no penalty. A smaller value poses a smaller threshold for the saccade amplitude to be penalized.

Here I modify these two hyperparameters. I try 2.0, 4.0, 6.0, 8.0 for α_{revisit} and 0.4, 0.6, 0.8 for $\alpha_{\text{amplitude}}$. This test is performed on the COCO-Search18 dataset. Note that I do not make the validation split. The agent learning rate is 0.001. The length of training is 30k. The training is performed three times independently. The measurement is the point distance, and the best checkpoint is used.

The results are presented in Figure 5.8. The error bars are the standard deviations of the three training repetitions. As shown, the fitting performance displays a relationship to the hyperparameters. For the revisit penalty, a smaller value can generate scanpaths more similar to humans, from the perspective of averaged point distance. For the amplitude threshold, a value close to 0.6 times the length of the diagonal of the search image produces the best result. This reveals that the human scanpaths may exhibit a pattern of revisitation and a certain saccade amplitude, and that the two environmental reward functions designed in MME indeed contribute to the scanpath modeling. By applying this test to comparable scanpath datasets, the behavior patterns of the corresponding subject groups can be compared. These patterns of revisitation and saccade amplitude may help with the diagnosis of certain types of mental disorders.

Note that the effect of the revisitation reward depends on the discretization method. A finer grid can reduce the occurrence of revisitations and change the effect of the revisitation penalty. Also, note that the current results display a high variance. Future work may perform more repetitions of training to obtain a more reliable result.

5.2.5 Ablation Tests

In this part, I check the effectiveness of the designs in my method. There are three core designs, including the decoupled reward function, the discounted score formulation, and the constraints on value ranges. In the implementation, some modifications are also made to the guided cost learning method, including the removal of human sample augmentation, the policy smoothing, and the greedy policy for sampling.

Among the three core designs, the decoupled reward function is the fundamental design for guaranteeing that a static reward map is learned. Without this design, there is no such guarantee. Also, the previous section 5.2.4 demonstrated that the designed environmental reward functions contribute to the performance. For the constraints on value ranges, I notice NaN errors in experiments caused by out-of-bound values when the constraints are removed. These errors hinder the experiment scripts from running. When these constraints are present, the experiments run normally and get the results such as those presented in Figure 5.7. This demonstrates the effectiveness of the constraints. In the following experiment, I test the effectiveness of the discounted score formulation, as well as the effectiveness of the removal of human sample augmentation. The other modifications, i.e., policy smoothing and greedy policy for sampling, are left as future work.

In this test, I select a single human scanpath from the ASD dataset that does not contain revisitations, so the default setting should be able to exactly replicate it. I use “agent score”

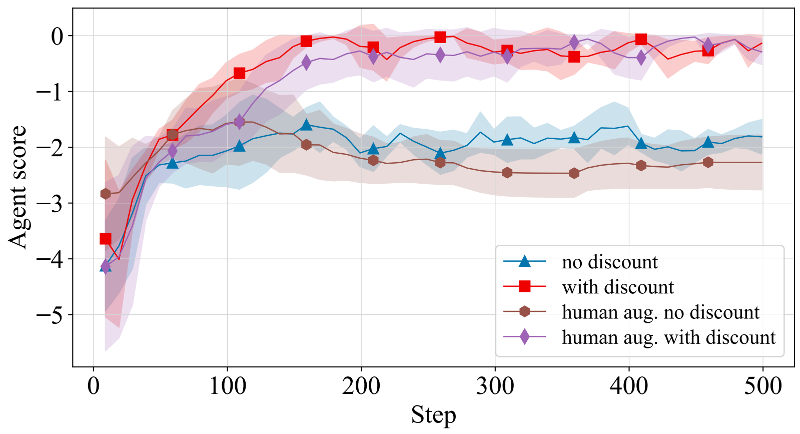


Figure 5.9: Learning curves of different variations of MME.

to measure the performance of the agent in training, which is the opposite value of the average point distance. Note that the point distance here is calculated by converting the fixations from pixels to grids according to Eq. (2.8). That is, the Euclidean distance between the discretized representations. Therefore, despite that the human fixations may not be at the center of the grids, this agent score can be zero when the agent fixates on the same grid as humans. Considering that the grid size is 128, the amplitude of the agent score is roughly 1/128 of the point distance in pixels.

The learning curves are plotted in Figure 5.9. The shaded regions in the figure show the standard deviation over 5 independent trainings. The curves are processed with exponential smoothing [195] with a smoothing factor of 0.5. The default MME is labeled as “with discount”. The variation without reward discount, i.e., $\gamma_r = 1$ is labeled as “no discount”. Also, the variants that augment agent samples with human scanpaths during training of the reward model are marked with “human aug.”.

As shown, the discounted score makes a significant contribution to the performance. The variants with discounting can converge to the optimal performance, while the variants without discounting cannot. For the scanpath augmentation, I notice that the final performance is not affected, but the learning speed is slower than the default MME, indicating that the removal of human scanpath augmentation improves the learning efficiency.

		Entropy Ratio		
		0.5	1	2
Learning Rate	0.0001	180.721	182.056	**179.286**
	0.00001	192.76	192.452	190.299

Figure 5.10: Hyperparameter tuning of SVPG on COCO-Search18 dataset.

5.2.6 Performance of SVPG

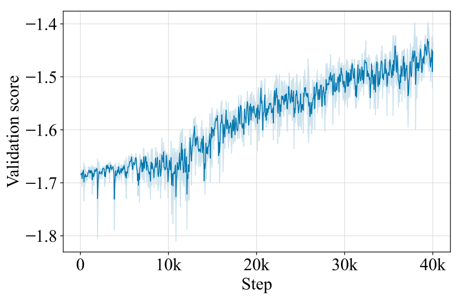
Finally, I examine the performance of SVPG on this visual search IRL task. As already mentioned, the length of the state vector is 32400, which is longer and more challenging than that in the other standard RL tasks. In addition, the reward functions are static in standard RL tasks but dynamic in IRL tasks, posing another challenge to training.

This test is done on the COCO-Search18 dataset. Since I aim to verify the ability of SVPG to replicate the human scanpaths, I train and validate it using the full 800 scanpaths, i.e., without a validation split. The training lasts 40k steps, and the model is validated every 50 steps. In validation, the agent acts greedily according to the policy distribution. The validation metric is the same as the "agent score" described before. A checkpoint corresponding to the best validation performance is saved and reported.

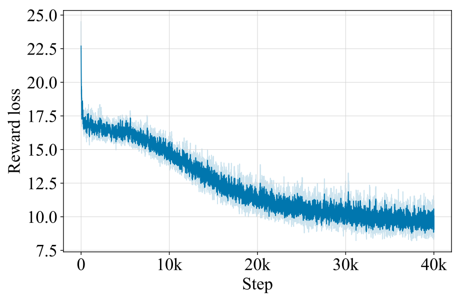
I tune the learning rate for the agent model and the entropy ratio in the PPO algorithm. The learning rate for the reward model is set to 0.0001. The mean performances, measured by pixel-wise point distance, over three independent trainings are reported in Figure 5.10. The best performance is marked with bold text.

The learning curves corresponding to the best setting are plotted in Figure 5.11, including the agent's validation performance and the reward loss. The shaded region shows the standard deviation value. The two curves are processed with exponential smoothing [195]. The reward loss curve uses a smoothing factor of 0.05, and the agent validation score curve uses 0.5.

As shown, the MME method, when implemented with the RWTA network and SVPG algorithm, can learn the visual search IRL task smoothly. Consider that the actions are discretized and the side length of the grids is 128, the current result indicates that, after training, the average difference between a human fixation and the corresponding prediction is less than twice the size of a grid.

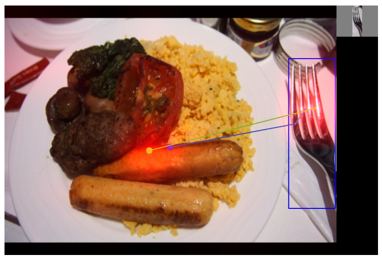


(a) Agent validation score.

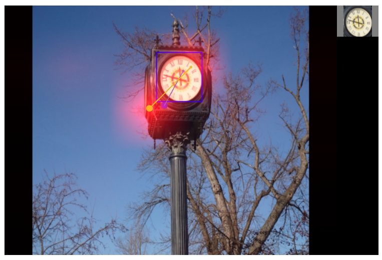


(b) Reward loss.

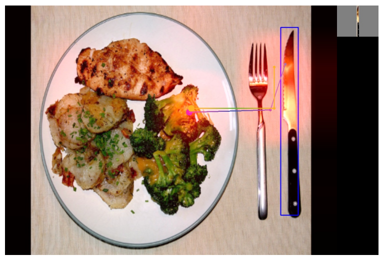
Figure 5.11: Learning curves of SVPG on COCO-Search18 dataset.



(a)



(b)



(c)

Figure 5.12: Visualizations of reward map, human scanpath, and a sampled agent scanpath on COCO-Search18.

Furthermore, I visualize the scanpaths of the agent and human. I randomly pick scanpath data from the dataset and plot the scanpaths, as shown in Figure 5.12 [4]. The target image is drawn to the upper right corner of the search image. The human scanpath and agent scanpath are respectively presented as blue and green lines, with the beginning marked by a big circle dot and subsequent fixations marked by small dots. Each reward map is normalized separately by dividing its maximum value, resized to the image size by nearest neighbor interpolation, and is shown in a red heatmap. A red region implies a high reward value.

As shown, the reward map and the agent's scanpaths capture the human scanpaths. The sequential information in the scanpaths is preserved in the agent. This shows the ability of SVPG to solve this IRL task.

The above successful application of SVPG to visual search scanpath modeling further demonstrates the capability of the RWTA network in representing complex functions and the effectiveness of the SVPG learning method. In addition, this is the first application of an R-STDP-based

[4]Figure 5.12a is adapted from “Emirates Business Class” by Isriya Paireepairit, used under CC BY-NC 2.0. Figure 5.12b is adapted from “clock” by Imbreathingyummyair, used under CC BY 2.0. Figure 5.12c is adapted from “Dinner 6-11-06” by Alvin Smith, used under CC BY-NC 2.0.

biologically plausible model to visual search. The brain-like model and learning rules make it possible to investigate the relationship between brain structure and brain function. For example, certain neurons in the network may be perturbed to test their effects on the visual search behavior.

5.3 Experiments on Social and Non-social Visual Search

Analysis of eye movement patterns of ASD can help improve the understanding of ASD and its detection techniques. In the previous sections, I designed MME, a new IRL method that can not only extract static reward maps from a set of human scanpaths but also compare the effect of different settings of environmental rewards. In this section, I perform a more thorough application of MME to the ASD dataset by training it on scanpaths from different types of subjects. Apart from MME, I also apply standard predictive models, including two behavior cloning methods, to extract other aspects of the behavior patterns. The contribution of this chapter can be viewed as an extension of the statistics-based analysis in [30]. As far as I know, this is the first attempt to use scanpath prediction models in ASD analysis.

5.3.1 Target-Conditioned Spatial Attention

I first look at the spatial attention extracted by MME. This is measured by applying the trained reward model to certain search images and target images. Following reference [30], I focus on scanpaths from four groups of subjects, namely people with amygdala lesions, ASD, ASD control group, and NUS control group. I use “amygdala”, “ASD”, “ASDctrl”, and “NUS” to refer to these subject groups. There are respectively 3, 13, 8, and 11 subjects in these groups. The numbers of available scanpaths, after pre-processing, are 469, 2067, 1236, and 1756.

I train an MME model for each subject group. There is no validation split. The models use the same default hyperparameters and are trained for 30k steps. The final checkpoint is used for the analysis. Because the numbers of scanpaths in each subject group are different, this approach ensures that the reward model and the agent model are trained for the same number of steps. A better approach may split the sub-dataset for each group into a training set and a validation set and get models that have the best validation performance. However, the amygdala group has a limited number of scanpaths, thus this can be challenging. A pre-trained feature extractor may help improve the generalization. This is left for future work.

To examine spatial attention, I apply the trained reward model to all the search images and target images. In the ASD dataset [30], there are 20 search images. Each search image contains 12 social objects and 12 non-social objects. All the objects are used as target images. This results in 480 combinations of search and target images, and thus there are 480 reward map predictions for each subject group. To facilitate analysis of image contents, I resize the reward map from the original size 8×6 to the size of the search image 1024×768. This is done by assigning the reward values to the pixels at the center of corresponding grids and then smoothing the map with a Gaussian kernel. The smoothing is implemented with the "GaussianBlur" function in OpenCV, and the kernel size is set to 513. For comparison between subject groups, the reward maps are normalized with respect to subjects so that the sum of reward values over search images and target images is the same across groups. For visualization, the smoothed reward map is divided by its maximum value and added to the red channel of the search image. Figure 5.13 presents two example reward maps obtained for a search image under two different targets. The target images are shown to the upper right corner of the search image.

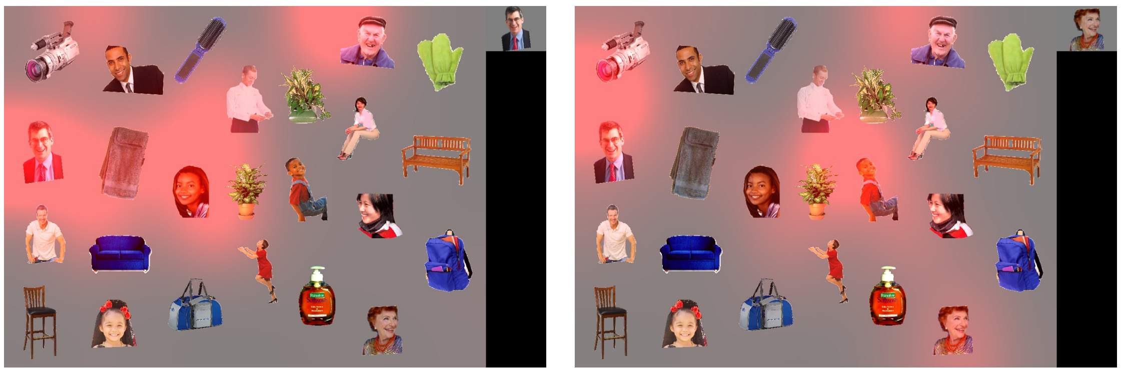
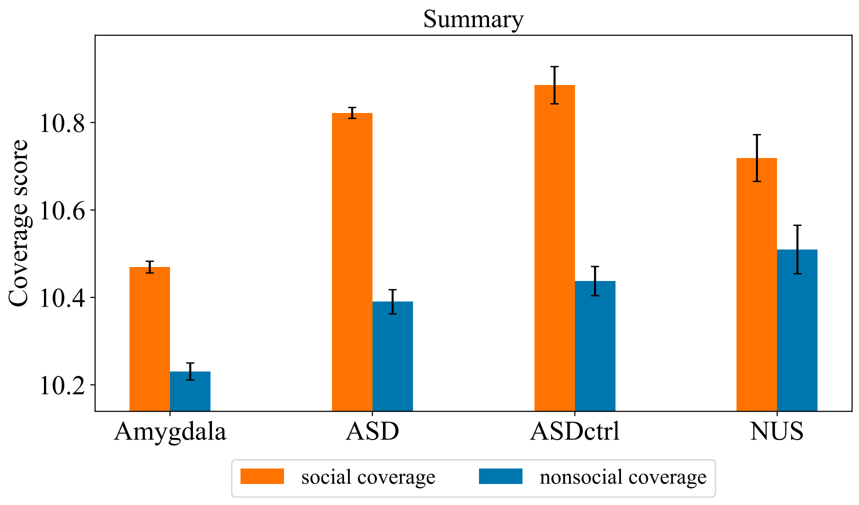

(a) (b)

Figure 5.13: Examples of reward map on a search image under different target images.

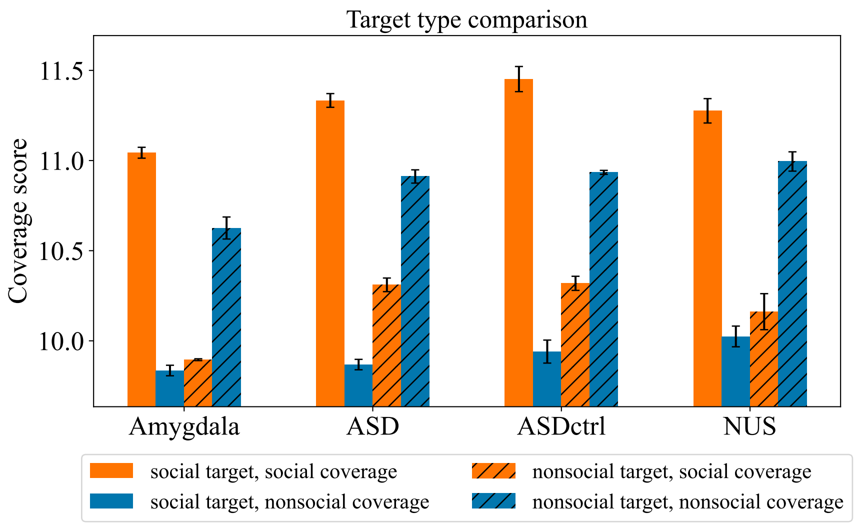
Note that, for each human subject, only a portion of the objects are presented as target images in the data collection process. An advantage of my predictive model is that it can be applied to unseen search or target images, making the testing data consistent across subject groups.

In addition, I emphasize that the reward map is different from a saliency map or initial fixation distribution. A saliency map captures the frequency of fixations on different areas, while a reward map reflects the priority of looking at different areas. An initial fixation distribution may display which part of the search image is more attractive, but it is subject to changes during the visual search process. In contrast, a reward map is static and is only conditioned on the environmental reward functions set during training.

I calculate the coverage of the smoothed reward map on different types of objects under different conditions to investigate the subject's spatial attention. The positions of the objects and the type (social or non-social) are provided in the ASD dataset. I use two maps to represent the positions of social and non-social objects. The maps are binary, with value 1 indicating objects and 0 otherwise. The coverage is calculated by multiplying the smoothed reward map by an object map and summing the values. It is then scaled by multiplying it by a positive constant for ease of presentation.



(a)



(b)

Figure 5.14: Coverage values across types of subjects and target images.

Figure 5.14a presents the coverage values of social objects and non-social objects for the four subject groups. Each setting is run three times, and the mean values are reported. The error bars show the standard deviation. A higher coverage value indicates larger reward values assigned to corresponding areas. As shown, for all the subject groups, the social objects are assigned higher reward values than non-social objects. This means that the social items attracted more attention than non-social items. This is consistent with the finding in [30] that neither people with ASD nor amygdala lesions have deficits in social preference. For the difference between groups, I find that the absolute coverage value does not differ significantly between the ASD group and the two control groups. In contrast, the amygdala group displays a significantly lower coverage value for both social and non-social objects. Considering that the sum of reward values is the same across groups, this means that more reward value is assigned to the background area for the amygdala group. This indicates a deficit of general attention to objects.

A further comparison of the spatial attention under social versus non-social target images is shown in Figure 5.14b. In this test, the reward models are evaluated on either social target images or non-social target images. The number of search-target image pairs is 240. As shown, all the subject groups exhibit a higher coverage value for objects of the same type as the target image. In addition, the preference for social objects when searching for a social target is stronger than in the case of non-social targets. This is consistent with the finding in [30]. However, my results do not show a significant difference between subject groups. In [30], the authors find that the ASD group shows reduced target-relevant effects at early fixations but not in later fixations. Considering that the higher values in my method correspond to early fixations, it is possible to extract the areas for early fixations from the reward map. Future work may take a closer look at the distribution of reward values in the reward maps.

5.3.2 Effects of Environment Variations

Next, I look at the effects of environmental variations on the performance of MME on scanpaths from different subject groups. In the previous section, I discovered a relationship between the two environmental reward functions and the training performance. Here, I additionally investigate the effects of perturbations on agent actions.

This part of the experiments trains one MME model for each subject group. There is no validation split for the dataset. The training lasts 30k steps, and a checkpoint is saved every 2k steps. For each metric, all the checkpoints are evaluated, and the best performance is reported.

Environmental Noise in Fixations

The environmental noise simulates randomness in the fixations. I implement this noise by perturbing the agent's fixation coordinates with two random variables independently sampled from the normal distribution. The strength of the noise is adjusted by multiplying the random variables with $\alpha_{\text{noise}} * \text{size}_{\text{search,diag}}$, where α_{noise} is a hyperparameter to be tuned.

I train MME models under different noise strengths from 0.0 to 0.6. For each setting, I repeat training three times and report the mean values. The error bars are the standard deviation values. The noise strength of 0.0 corresponds to the default setting, i.e., no noise added. Note that the environmental noise is added to both training and testing.

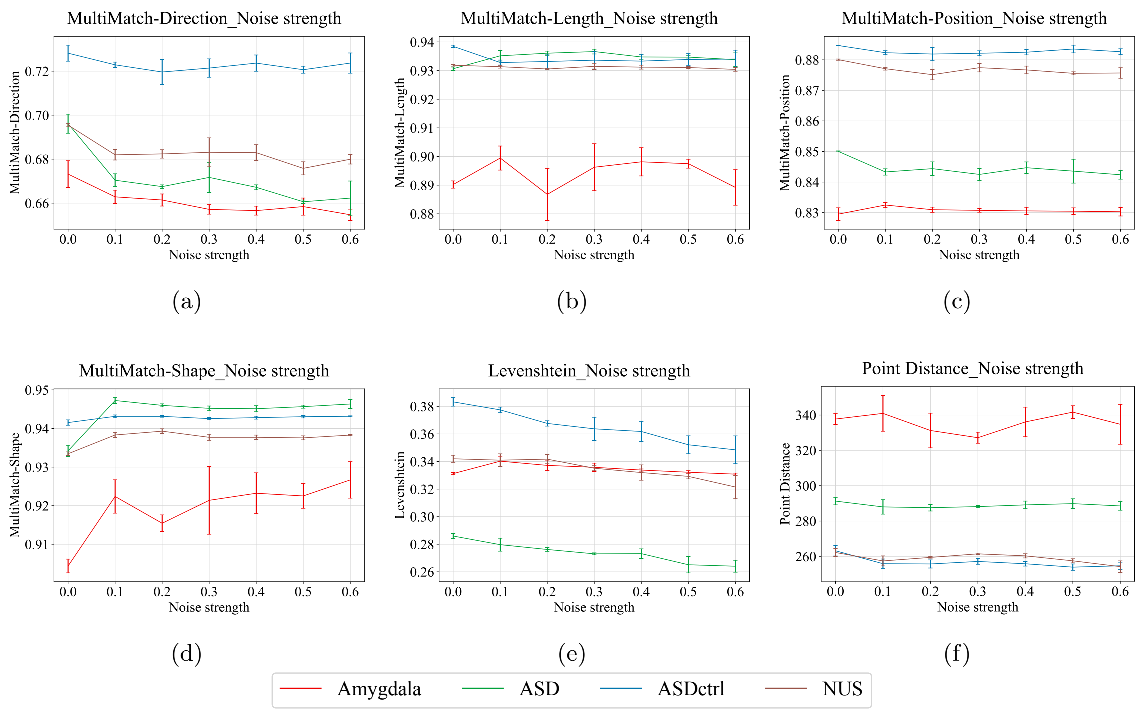


Figure 5.15: Effect of environmental noise on performances.

Figure 5.15 presents the results under six metrics. It can be inferred that:

- Under all the MultiMatch metrics, the amygdala group generates the lowest score. This indicates that the corresponding scanpaths are more difficult to replicate by the MME model.

- Under MultiMatch metrics, the two control groups do not exhibit significant changes in performance when the noise strength changes. When the noise strength increases, the ASD group shows a decrease in MultiMatch-direction, and the amygdala group shows an increase in MultiMatch-shape. Considering that MultiMatch-shape captures the vector differences and that MultiMatch-direction measures the angular distance, my results imply that (1) the direction of saccades from the ASD group may be less random than the control groups, so that add noise brings more harm to the performance, and (2) the shape of scanpaths from the amygdala group may be more noisy than the other groups.
- Similarly, under the Levenshtein similarity, I observe that the performances of ASD and the two control groups gradually decrease as the noise strength increases. This is normal because a stronger noise can make the agent deviate from the expected scanpath with a higher probability. In contrast, the amygdala group does not exhibit a significant performance decrease. This implies a stronger inherent randomness in the fixations of the amygdala group.

Revisitation Penalty

In MME, the revisitation penalty is part of the additional environmental reward. A higher penalty better prevents the agent from fixating on fixated points in the current scanpath. When there are no revisitations in human scanpaths, a higher revisitation penalty may help accelerate training. When there are many revisitations in human scanpaths, a high revisitation penalty will hinder the agent from replicating human behavior. In this test, I test the choice of α_{revisit} from 0, 2, 4, 6, 8, 10. The value 10 is the default setting, which, together with regularization, makes the combined reward always negative for revisitations. The training procedure is the same as the previous test of environmental noise.

Figure 5.16 presents the performances under six metrics. The results under the point distance metric show that all subject groups show a slight performance increase when the revisitation penalty is lowered. This means revisitations exist in human scanpaths, and that allowing revisitations in agent scanpaths can make their fixations closer to humans. For the other metrics, significant results exist for the ASD group. With a smaller revisitation penalty, the MultiMatch-shape performance increases, and the MultiMatch-direction performance decreases. The changes in the performances of other groups are not significant. This shows the outstanding sensitivity

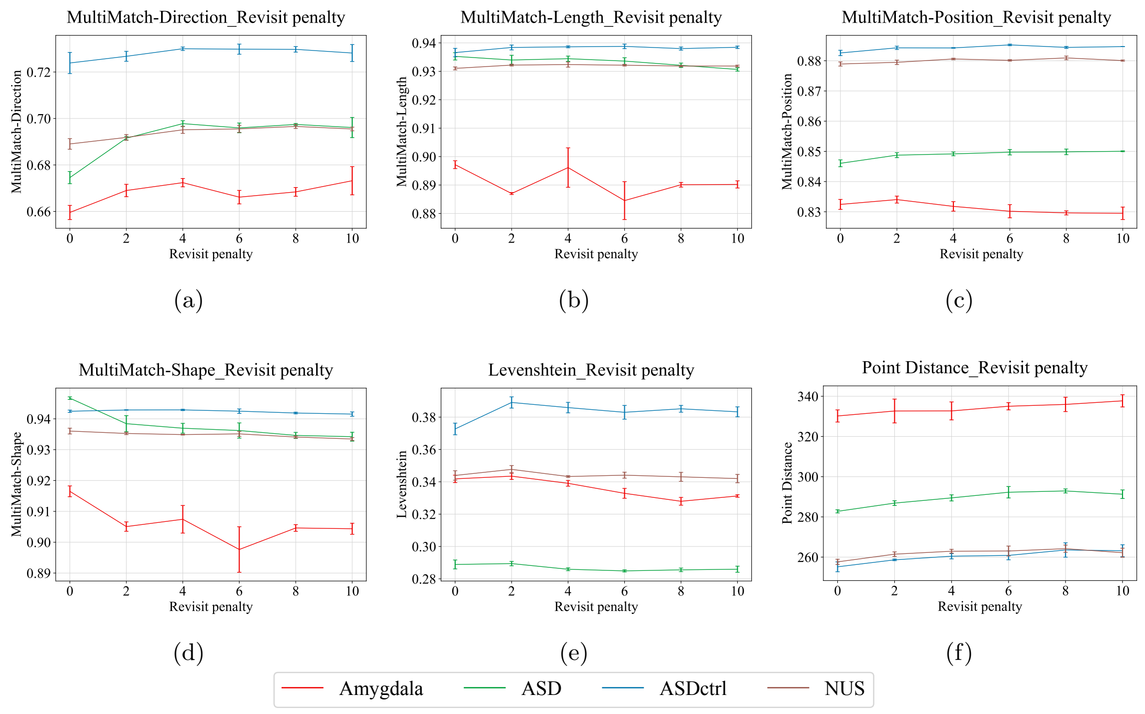


Figure 5.16: Effect of revisitation penalty on performances.

of the ASD group to the revisitation penalty. Revisitations in scanpaths from the ASD group contribute to the scanpath shape while also making it more difficult to replicate the saccade directions.

Saccade Amplitude Penalty

The saccade amplitude penalty is another environmental reward. A threshold hyperparameter $\alpha_{\text{amplitude}}$ of value 1 means no penalty. A smaller value will encourage the agent to take short saccades. Therefore, if human scanpaths contain many long saccades, a small $\alpha_{\text{amplitude}}$ will likely decrease the performance.

The histogram of saccade lengths can be explicitly calculated. The contribution of this test is to understand "what performance can be obtained in imitating humans when the agent is constrained to make short saccades". This is a combination of partial observation, other sources of reward, and similarity metrics.

I test the choice of $\alpha_{\text{amplitude}}$ from 1.0, 0.8, 0.7, 0.6, 0.5, 0.4, and 0.3. The value 1.0 is the default setting. The training procedure is the same as the previous test of environmental noise.

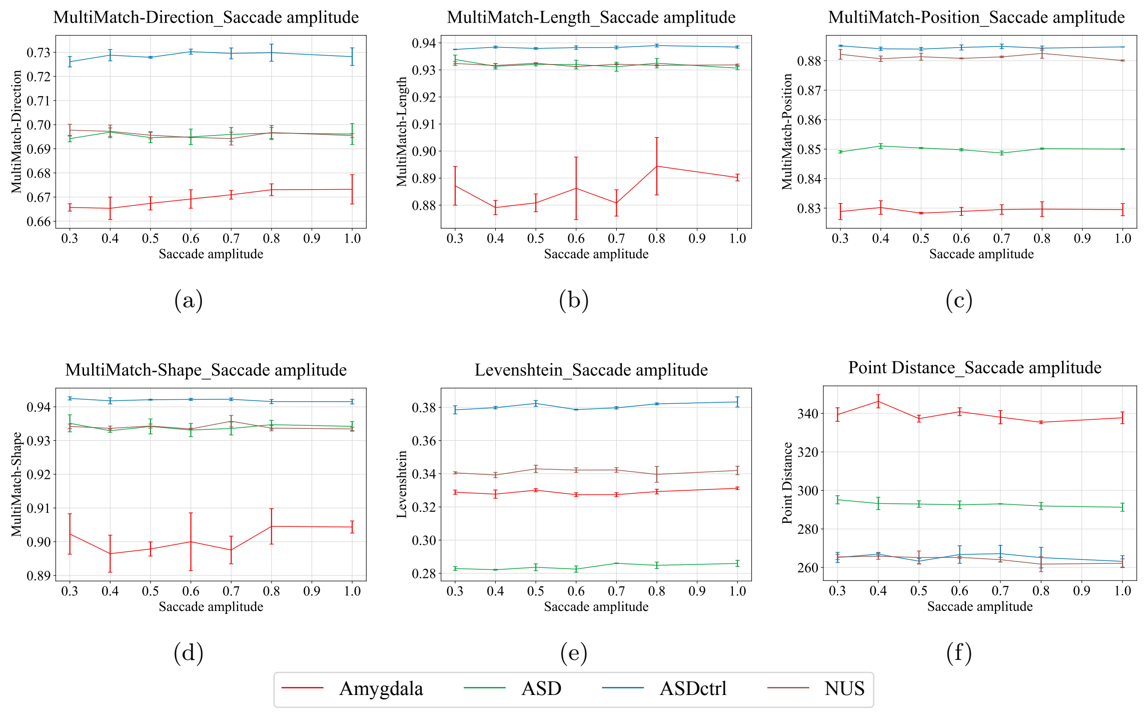


Figure 5.17: Effect of threshold for saccade penalty on performances.

Figure 5.17 presents the results. For the range of hyperparameters I test, the ASD, ASDctrl, and NUS groups do not show significant differences in performance under all the metrics considered, meaning these human scanpaths can be well simulated with short saccades. In contrast, the performance of the amygdala group is more affected. A larger threshold generates a better performance under the MultiMatch-direction metric, indicating a contribution of long saccades to better capture the saccade directions. Although less clear, trends are evident for the MultiMatch-length and MultiMatch-shape metrics.

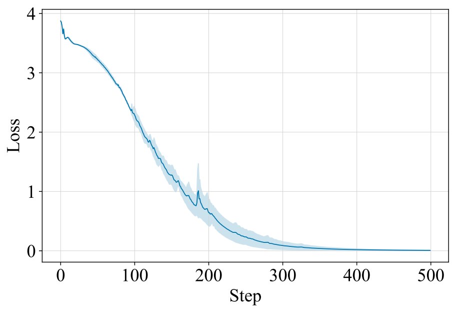
In summary, the tests in this section demonstrate outstanding properties of scanpaths from the ASD group and the amygdala group. The amygdala group displays sensitivity to action noises and saccade amplitude under certain metrics, while the ASD group is sensitive to action noises and revisitation penalty under certain metrics.

5.3.3 Memory Load

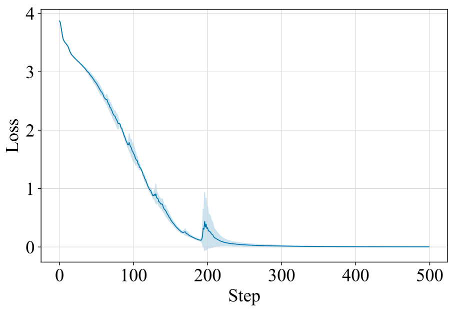
In reference [30], a test of the effects of cognitive load is performed by comparing the target-relevant effects on tasks with different numbers of objects in the search images. In this study, I focus on the memory load and take a more direct and economic method by training predictive models with different memory capacities and comparing their performances.

I use a behavior cloning model implemented with an LSTM network. The network consists of some convolutional layers for the processing of the partial observation and the target image, and an LSTM layer for capturing historical information. The features extracted by the convolutional layers are concatenated together and input to the LSTM layer, followed by some fully connected layers that output the action distribution. I change the size of the LSTM layer to modify the memory capacity of the network for historical information. A larger size is able to capture more information.

In this test, I split the scanpaths according to the target image type and subjects and train one model for each of them. Considering the small number of available scanpaths for each subject, there is no validation split in the dataset. All the models are trained for the same number of 500 steps. The final checkpoint is used for evaluation. For the size of the LSTM layer, I test the choices of 25, 50, 100, 200, and 400.



(a) LSTM size 25.



(b) LSTM size 400.

Figure 5.18: BC-LSTM. Learning curves with different sizes of the LSTM layer.

Figure 5.18 presents the learning curves on the scanpaths targeting non-social objects from the 30th subject. The curves are obtained from LSTM layer sizes of 25 and 400, representing the two extreme cases. The curves display the loss value, and the shaded regions show the standard deviation from 3 independent runs. The curves are smoothed in the same way as for Figure 5.11. As shown, the 500 number of steps is appropriate to make the training converge.

I then look at the performances. Figure 5.19 presents the mean performance values with respect to the LSTM sizes. The curve labels indicate the subject group and the type of target images. The values are averaged across scanpaths, subjects, and 10 independent trainings. As shown, there is a consistent increase in performance value when the LSTM size increases, regardless of the subject group, target type, and the performance metric. This shows that historical

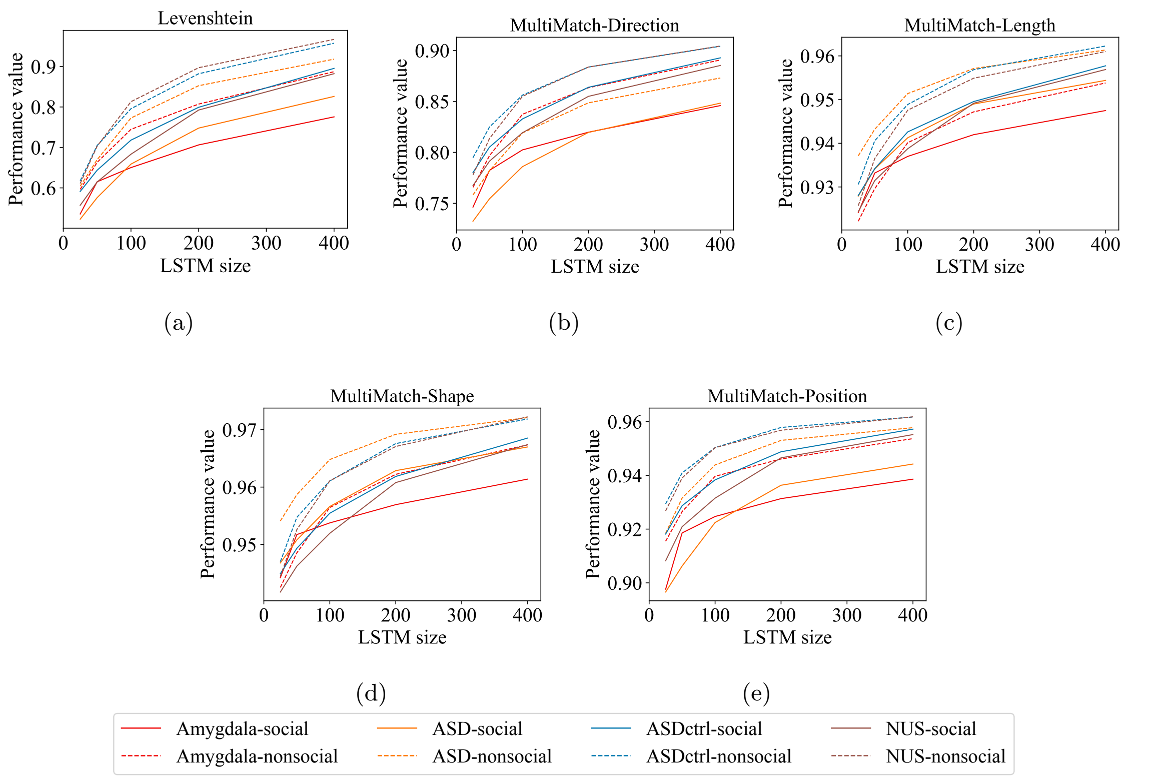
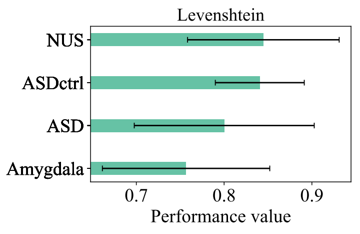


Figure 5.19: BC-LSTM. Comparison of performances with different sizes of the LSTM layer.

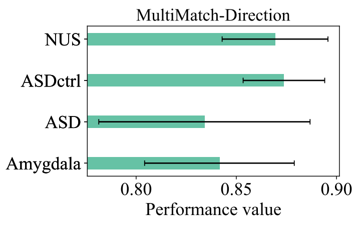
information is critical to replicating human scanpaths. Additionally, the increase in performance is weaker for larger LSTM sizes, meaning that there may be a threshold for size that just captures all historical information.

For a clearer comparison, in Figure 5.20, I plot the performances for the setting where the LSTM size is 200. The bar lengths are the mean values over training repetitions, subjects, and scanpaths. The error bars are the deviation over subjects. As shown, the two control groups exhibit similar performances. The ASD group has lower performance under the Levenshtein, MultiMatch-direction, and MultiMatch-position metrics. The amygdala group has lower performance under all the metrics. Considering that increasing the LSTM size improves the performance of all the groups, these results imply that a larger LSTM size is needed for the ASD and amygdala groups to obtain the same level of performance as the control group. That is, the scanpaths from the amygdala group display a larger memory load than the ASD group and the control group.

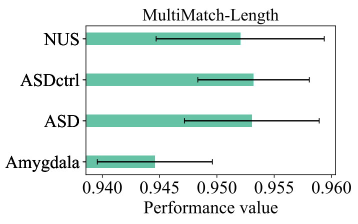
Figure 5.20 displays large standard deviations in performance values. To understand the source of the deviations, I calculate the deviations over the repetitions of training. Example results



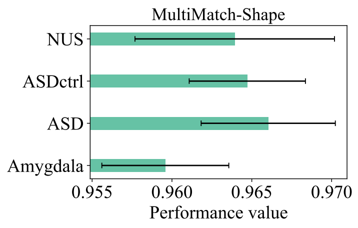
(a)



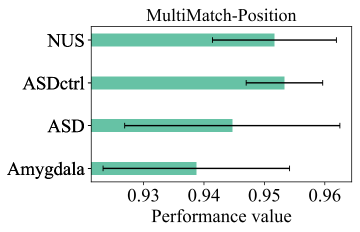
(b)



(c)



(d)



(e)

Figure 5.20: Comparison of performances on data from different types of subjects.

are in Table 5.2. As shown, the standard deviations among repetitions are much lower than in subjects. This shows that the deviations mainly come from the subject differences.

Table 5.2: Comparison of standard deviation values over subject groups and training repetitions.

Group-Metric	Mean	Std-subject	Std-repetition
NUS- Levenshtein	0.8447	0.0864	0.0101
Amygdala-MultiMatch-Length	0.9446	0.0050	0.0024
ASD-Point Distance	128.5935	43.3348	1.9948

A further inspection of the target image types is shown in Figure 5.21. For convenience, I use straight lines to mark the social-non-social pairs for all subject groups and LSTM sizes. As shown, for the ASD group and the control groups, the scanpaths under social targets have a consistently lower performance, meaning that social targets drive a search process that makes more use of historical information. For the amygdala group, there are a few discrepancies under the MultiMatch-shape and the MultiMatch-length metrics. This may indicate an abnormality in the balance between social and nonsocial targets, but may also be due to noise in data and training.

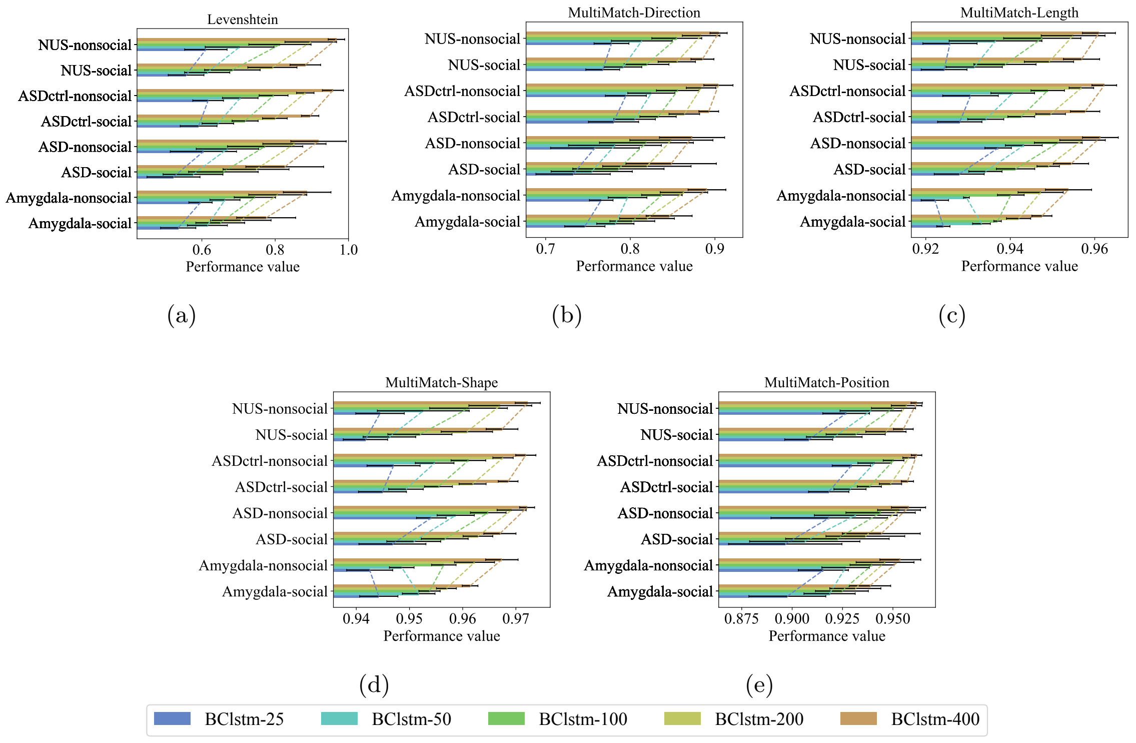


Figure 5.21: Comparison of performances on data collected with different types of target images.

5.3.4 Temporal Attention

Another perspective of memory is the temporal attention, which is concerned with the importance of information at different time-steps. I use the BCtransformer method in the previous section to extract attention maps from the sequence of recent histories.

The datasets are prepared in the same way as in section 5.3.3. A BCtransformer model is trained for each subject and social/non-social type of targets. The Transformer I use has three attention blocks. To extract the attention map, I multiply the attention weights from the attention blocks sequentially. The attention corresponding to the output is used, which reflects the importance of each historical observation to the fixation decision. In my setting, the history length is 10, so the attention map has 10 values. I normalize each attention map by dividing it by the attention value for the most recent step, so the process values represent the relative importance with respect to the current observation. I average the attention maps over fixations and subjects to obtain a summary. Considering that the observation history is padded with placeholders for early fixations, I summarize the attention maps separately for each length of valid history. For example, the attention maps of fixations that are at the 6th step of a scanpath are treated as one group. The attention maps for fixations at the 10th step or later are treated as one group because their lengths of valid history are all 10.

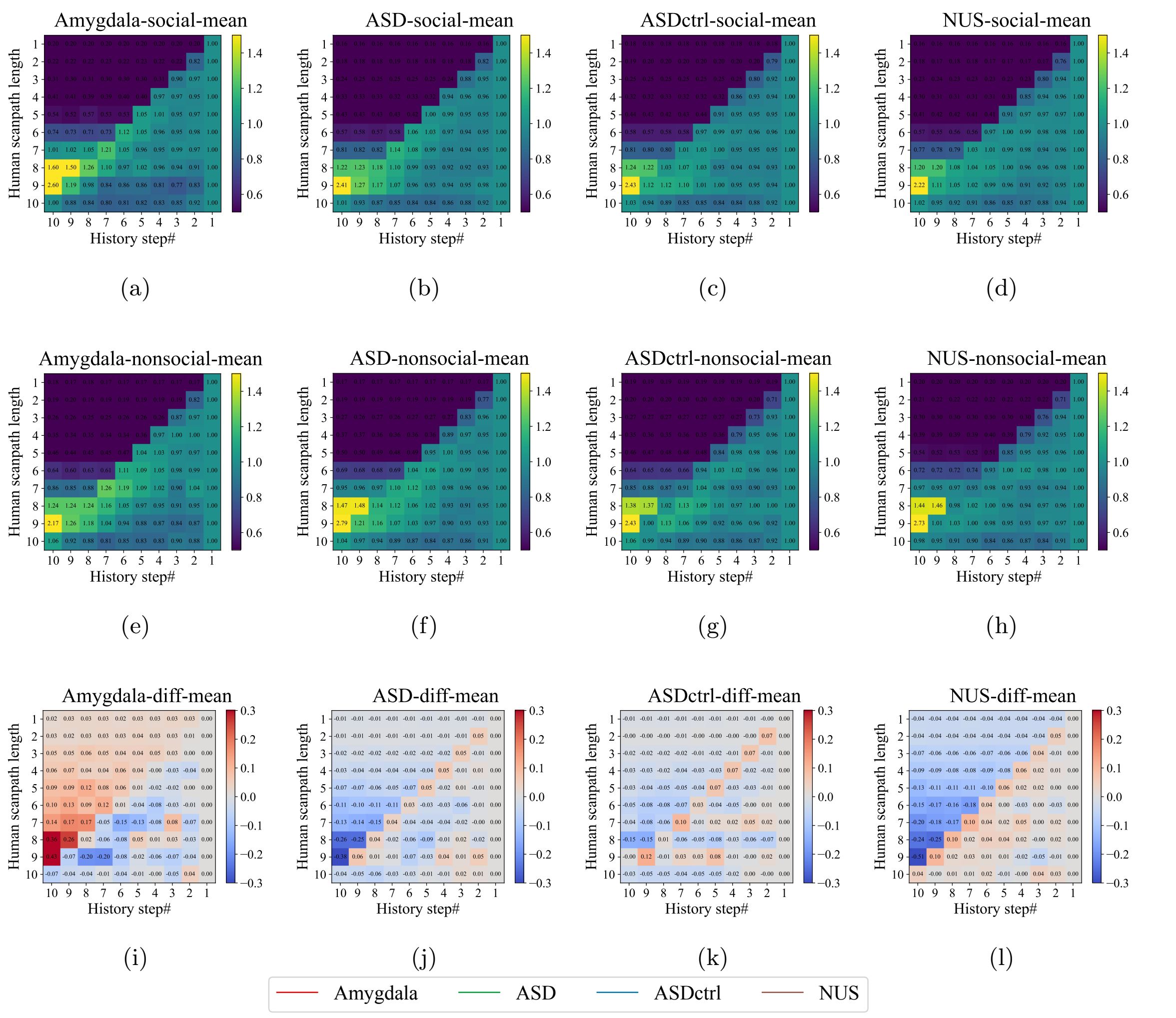


Figure 5.22: Visualizations of importance values for historical observations. Upper row: Social target. Middle row: Non-social target. Lower row: Difference. Columns: Amygdala, ASD, ASDctrl, and NUS.

This test is repeated 10 times, and the mean values are plotted in Figure 5.22. For each plot in Figure 5.22(a)-(h), the attention maps are plotted horizontally. The first row shows the attention map for fixations at the first step of a scanpath. The values, from left to right, are the normalized importance values for old to new observations. The rightmost value corresponds to the latest observation, so it has a value of 1 after normalization. The other values are smaller than 1, which is normal because they correspond to placeholders in the history. Similarly, the rows, from top to bottom, respectively show the attention map for fixations at the second to ninth steps and the rest steps in a scanpath. For each plot, the lower right area is effective.

Table 5.3: Average importance value for the first observation in history.

Subject type	Amygdala	ASD	ASDctrl	NUS
Social target	1.051	1.030	0.951	0.952
Non-social target	1.062	0.997	0.901	0.894

The columns of Figure 5.22 show results from the amygdala, ASD, ASDctrl, and NUS control groups. The first row (a-d) shows the results on scanpaths with social target images. The second row (e-h) shows the results for non-social target images. To better show the difference in temporal attention between social and non-social targets, the differences between the social results and the non-social results are shown in the third row (i-l). For ease of comparison, the figures in each row share the same color map.

Based on Figure 5.22, the following discussions can be made:

- In plots (a-h), there is a dark area in the middle of the lower right part. This shows that larger importance values are assigned to the early and late observations in the history. This is consistent for all subject groups and target image types.
- There is a difference between subject groups in the importance values for early observations in history. Specifically, for the first observation in history, I average their normalized importance values (which are located at the anti-diagonal of the plots) and present them in Table 5.3. As shown, the control groups have similar values, while the ASD and amygdala groups have larger values. This indicates that the scanpaths from the ASD group and the amygdala group depend more on the first observations in history.
- The importance values for the first observations are generally higher under social target images than non-social ones. This holds for the ASD, ASDctrl, and NUS groups, indicating that social targets demand more attention to early observations. This does not hold for the amygdala group, where the values for nonsocial settings are similar to those for social settings. This may reveal excessive attention to early observations under non-social targets in the amygdala group.

5.4 Chapter Summary

In this chapter, I applied the RL method to visual search modeling tasks. I developed the MME method for extracting an agent-independent reward map for a better comparison of behavior motivations. Results showed that MME can achieve a comparable performance to standard methods, and the environmental reward function in MME has the potential to reflect properties of visual search behavior. The application of SVPG showed that SVPG can solve the learning task under MME and replicate human behaviors.

Furthermore, I applied the MME method to the analysis of spatial attention, saccade amplitude, revisitation, and randomness of the ASD dataset. The reward maps extracted by MME were used to calculate the accumulated coverage of social and non-social objects under social and non-social target images. The environmental reward functions in MME were tuned to compare their effects on the learning performance of different subject groups. Some settings obtained different results for the ASD and amygdala group than the control groups, meanwhile obtaining similar results for the two control groups, which validates the comparison. I also included two tests based on behavior cloning models to analyze temporal attention and memory load. Some of my findings are consistent with previous studies, while others provide new insights into the visual search behavior patterns.

Chapter 6

Conclusion and Discussion

6.1 Conclusion

In this thesis, I studied the design of novel network models and learning methods for RL tasks. In addition to examining my models and methods on standard benchmark RL tasks, I also applied the proposed techniques to real-life human visual search behaviors to understand the cognitive mechanisms of visual search. The models and methods proposed in this thesis are inspired by experimental findings from neuroscience. In return, I explored using my proposed methodology to study human visual behaviors for both understanding the mechanisms of cognitive behaviors and revealing potential biomarkers for certain brain disorders.

First, I proposed SVPG, a new R-STDP-based SNN RL method. SVPG is designed for the RWTA network and has the advantage of being biologically plausible. Experiments show that SVPG can solve a series of standard RL tasks and an IRL-based visual search modeling task. SVPG also provides better inherent robustness to some types of input noise, network parameter noise, and environmental variations in the inverted pendulum task. For computation costs, SVPG is slower than BP in inference on GPU and faster in optimization. The memory cost and sample efficiency of SVPG are similar to BP, the most popular method for training ANNs. All parts of the RWTA network contribute to the performance. Other properties of SVPG include less selectivity in hidden parts than the output after training, and relative sparsity in weights compared to BP. Since SVPG only relies on the network, it can be combined with a wide range of existing RL techniques, including environment augmentation, reward engineering, etc.

For the real-life visual search task, I additionally proposed a new IRL method, MME, that features decoupling the reward function representation into an agent-independent reward map and some stimulus-independent environmental reward functions. MME has the advantage of facilitating the comparison of the attractiveness of different parts of the stimulus. Experiments show that MME can achieve comparable performances to standard methods on standard datasets. The core designs in MME are shown to contribute to the performance. The environmental reward functions in MME facilitate the investigation of high-level properties of scanpaths. Application of MME to an ASD dataset confirms some of the previous findings, while also providing some new insights about the eye movement behaviors of people with ASD or amygdala lesions.

The success of SVPG in solving challenging RL tasks indicates that R-STDP methods, although constrained to local learning rules, are capable of achieving comparable performances to standard backpropagation-based methods. This also reveals the potential of biologically plausible methods. In this thesis, the key to scaling up the R-STDP methods is the network structure. By upgrading a three-layer network to a fully-connected network, the performance increased. This, to some extent, conforms to the trend in the field of deep learning, where network structures such as convolutional layers [4] and attention blocks [6] bring improvements to the overall performance. In addition, my results further demonstrate the effectiveness of variational inference, as adopted by existing works [47, 64], in deriving R-STDP learning rules for SNNs.

In the application to visual search, a main finding is that human scanpaths, as represented by the three visual search datasets used in this thesis, can be modeled by two separate drives, i.e., a fixation-independent motivation map, and stimuli-independent environment rules (amplitude penalty and revisitation penalty). It is based on this finding that my method can be applied to extract motivation maps for understanding social and non-social visual search patterns. In addition, the successful application of SVPG to this task indicates that the visual search behavior can be replicated by a biologically plausible model. Further, in the analysis of the ASD dataset, differences in the experiment and control groups are noticed, which shows the effectiveness of the static motivation representations extracted by my method.

6.2 Limitations and Future Work

Here I summarize the limitations of the methods and experiments in this thesis and discuss the potential future work. For clarity, they are grouped into the SVPG part, the visual search part, and the analysis of social and non-social behavior.

In the design of the RWTA network and SVPG method:

- **Neuron model.** In the selection of the excitatory postsynaptic potential κ, I used the rectangle function due to its simplicity. Future work may consider the double exponential function, which is more common in the literature [110, 111]. For the encoding method, I used the rate coding due to its popularity. Other encoding methods such as temporal coding may preserve more details in the spike trains. Future work may investigate how to adapt the SVPG method to other types of encoding methods.

- **Network topology.** The current design of the RWTA network has a fixed topology. Previous works have investigated methods for evolving the structure [119, 120] to obtain better topologies in training. Future work may investigate this direction to further improve the task performance. In addition, I assumed that the action space of the RL task contains a finite number of actions. This makes it possible to assign an action neuron for each action. However, this design of the RWTA network does not suit RL tasks with a continuous action space. Future work may investigate adaptations, such as parameterized probability distributions, to these tasks.

- **Learning method.** SVPG is slightly beyond the definition of STDP, in which W_{pre} and W_{post} are constants with reference to neuronal activities. In SVPG, they are replaced by v_i and v_j, which are neuronal activities. Nevertheless, SVPG is only based on local signals, so it is more biologically plausible than backpropagation-based methods. In addition, the PPO-based implementation is not fully biologically plausible. This is due to the policy checkpoint required by the PPO algorithm to constrain the size of optimization steps. The checkpoint contains information beyond local signals. Note that this limitation is specific to the base RL algorithms. The REINFORCE-based implementation does not have this limitation.

- **Hardware implementation.** Experiments in this thesis indicate that SVPG has a slow inference speed. Since SNNs are event-based, the implementation on neuromorphic hardware is a promising direction for improving the speed. It will also reveal the energy consumption of the SVPG method.

- **Cause of robustness.** In my experiments, I observed that SVPG produces a better robustness to various types of perturbations than the other methods. Although the random firing process and the noisy hidden WTA circuits are proposed as possible explanations, the true reason for the advantage in robustness is unknown. Future work may have a deeper look at the network and training process to investigate how the robustness is produced.

In the design of the IRL method for visual search scanpath modeling:

- **Visual search task modeling.** In this thesis, the scanpath prediction task forces the agent to carry out a scanpath with the same length as humans. That is, the termination of the visual search process is excluded from the prediction task. Additionally, the durations of fixations are also neglected. Future work may take this information into consideration and build a more comprehensive scanpath prediction model, which may offer more insights about behavior patterns. For the partial observation of the agent, I used the cropped portions of the original search images. This simple setting can be easily applied to different datasets, but also makes the dimension of the state space high. Due to the small size of visual search datasets, the generalization to the testing set is difficult. Previous studies often use a feature extractor to process the search image [55, 60], which may improve the generalization of the trained models. Future work may adopt feature extractors to improve the prediction performance. For the space of fixation actions, the current model uses a coarse grid of action space. Making the grid finer may change the effects of the environmental reward function such as the revisitation penalty, and change the task performance of MME. Considering that a larger grid size, or no discretization at all, leads to a large action space and thus makes training more difficult, future work may also introduce parameterized distributions to represent the agent policy.

- **Ablation study.** In my design of the IRL method, I adopted the guided cost learning method [161] and removed one of its regularizations that aims to reduce high-frequency variations. Future work may investigate its effect on learning performance. Also, the effects of two minor designs, i.e., policy smoothing and greedy policy for sampling, can be investigated in future work.

- **Experiments.** In the experiments, a shrunk version of the COCO-Search18 and the IVSN datasets was used to save computation cost. Future work may test the methods on the full set of these datasets to get more accurate results. In the implementation of baseline methods such as GAIL, the default hyperparameters were used in training. This may not generate the best possible performance on certain datasets. Future work may try different hyperparameters to get a more comprehensive understanding of the performances. In addition, current results are based on three repetitions of training. Considering the noise in task performances, more repetitions can help generate more reliable results.
- **Applications.** This thesis applies the visual search model to pattern analysis of ASD-related disorders. Since the motivation extraction does not rely on this specific scenario, it is possible for future work to apply my proposed technique to other scenarios such as driver assistance.

In the analysis of social and non-social visual search behavior patterns:

- When generating the motivation representations, I use the final checkpoint of MME in training. Although this makes a fair comparison, the generalization of the model on the scanpath data can be different for each subject group. When a subject group has a small number of scanpaths, the corresponding model may not generalize well to the unseen search and target images, thus affecting the comparison results. Considering the small number of scanpaths available, future work may adopt techniques that help improve the generalization. For example, using a pre-trained feature extractor to reduce the dimensionality of the observations, and pre-training the models on the entire scanpath dataset before fine-tuning them on the specific subject groups.
- In the investigation of target-conditioned spatial attention, I did not distinguish early and late fixations in scanpaths. My results do not show a significant difference between subject groups, which is different from [30], where the authors find that the ASD group shows reduced target-relevant effects at early fixations but not in later fixations. Future work may take a closer look at the temporal distribution of reward values in the reward maps to have a deeper understanding of this behavioral pattern.

Bibliography

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT press, 2018.

[2] C. J. C. H. Watkins and P. Dayan, “Q-Learning,” *Machine Learning*, vol. 8, pp. 279–292, 1992.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” *arXiv:1312.5602*, Dec. 2013.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[5] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-Level Control Through Deep Reinforcement Learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[8] G. Lample and D. S. Chaplot, “Playing FPS Games with Deep Reinforcement Learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017.

[9] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, “Learning Dexterous in-Hand Manipulation,” *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, Jan. 2020.

[10] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, “Mastering the Game of Go Without Human Knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[11] L. Wu, F. Tian, T. Qin, J. Lai, and T.-Y. Liu, "A Study of Reinforcement Learning for Neural Machine Translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3612–3621.

[12] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep Reinforcement Learning with Successor Features for Navigation Across Similar Environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 2371–2378.

[13] H. Gao, Z. Yang, X. Su, T. Tan, and F. Chen, "Adaptability Preserving Domain Decomposition for Stabilizing Sim2Real Reinforcement Learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2020, pp. 4403–4410.

[14] H. Li, Q. Zhang, and D. Zhao, "Deep Reinforcement Learning-Based Automatic Exploration for Navigation in Unknown Environment," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 2064–2076, 2019.

[15] H. Quan, Y. Li, and Y. Zhang, "A Novel Mobile Robot Navigation Method Based on Deep Reinforcement Learning," *International Journal of Advanced Robotic Systems*, vol. 17, no. 3, May 2020.

[16] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A Survey of Deep Reinforcement Learning in Video Games," *arXiv:1912.10944*, Dec. 2019.

[17] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, "A Survey of Robotics Control Based on Learning-Inspired Spiking Neural Networks," *Frontiers Neurorobotics*, vol. 12, 2018, art. no. 35.

[18] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven Exploration by Self-supervised Prediction," in *Proceedings of the 34th International Conference on Machine Learning.* PMLR, 2017, pp. 2778–2787.

[19] P.-L. Bacon, J. Harb, and D. Precup, "The Option-Critic Architecture," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 1726–1734.

[20] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2017, pp. 23–30.

[21] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning," in *International Conference on Learning Representations*, 2019.

[22] K. Wang, B. Kang, J. Shao, and J. Feng, "Improving Generalization in Reinforcement Learning with Mixture Regularization," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 7968–7978.

[23] C. Tessler, Y. Efroni, and S. Mannor, "Action Robust Reinforcement Learning and Applications in Continuous Control," in *Proceedings of the 36th International Conference on Machine Learning.* PMLR, May 2019, pp. 6215–6224.

[24] Z. Liu, J. Lu, J. Xuan, and G. Zhang, "Deep Reinforcement Learning in Nonstationary Environments With Unknown Change Points," *IEEE Transactions on Cybernetics*, pp. 5191–5204, 2024.

[25] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," in *Proceedings of The 33rd International Conference on Machine Learning.* PMLR, Jun. 2016, pp. 1928–1937.

[26] M. Chahine, R. Hasani, P. Kao, A. Ray, R. Shubert, M. Lechner, A. Amini, and D. Rus, "Robust Flight Navigation Out of Distribution with Liquid Neural Networks," *Science Robotics*, vol. 8, no. 77, p. eadc8892, Apr. 2023.

[27] J. Wu, C. Xu, X. Han, D. Zhou, M. Zhang, H. Li, and K. C. Tan, "Progressive Tandem Learning for Pattern Recognition With Deep Spiking Neural Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7824–7840, Nov. 2022.

[28] A. Suhaimi, A. W. H. Lim, X. W. Chia, C. Li, and H. Makino, "Representation Learning in the Artificial and Biological Neural Networks Underlying Sensorimotor Integration," *Science Advances*, vol. 8, no. 22, p. eabn0984, Jun. 2022.

[29] Y. Zeng, Y. Zhao, T. Zhang, D. Zhao, F. Zhao, and E. Lu, "A Brain-Inspired Model of Theory of Mind," *Frontiers in Neurorobotics*, vol. 14, Aug. 2020, art. no. 60.

[30] S. Wang, J. Xu, M. Jiang, Q. Zhao, R. Hurlemann, and R. Adolphs, "Autism Spectrum Disorder, but Not Amygdala Lesions, Impairs Social Attention in Visual Search," *Neuropsychologia*, vol. 63, pp. 259–274, Oct. 2014.

[31] D. Patel, H. Hazan, D. J. Saunders, H. T. Siegelmann, and R. Kozma, "Improved Robustness of Reinforcement Learning Policies Upon Conversion to Spiking Neuronal Network Platforms Applied to Atari Breakout Game," *Neural Networks*, vol. 120, pp. 108–115, Dec. 2019.

[32] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "A Solution to the Learning Dilemma for Recurrent Networks of Spiking Neurons," *Nature Communications*, vol. 11, no. 1, Dec. 2020, art. no. 3625.

[33] A. Yanguas-Gil, "Coarse Scale Representation of Spiking Neural Networks: Backpropagation Through Spikes and Application to Neuromorphic Hardware," in *International Conference on Neuromorphic Systems.* ACM, Jul. 2020, pp. 1–7.

[34] N. Frémaux and W. Gerstner, "Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules," *Frontiers in Neural Circuits*, vol. 9, Jan. 2016, art. no. 85.

[35] M. Yuan, X. Wu, R. Yan, and H. Tang, "Reinforcement Learning in Spiking Neural Networks with Stochastic and Deterministic Synapses," *Neural Computation*, vol. 31, no. 12, pp. 2368–2389, 2019.

[36] G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "Biologically Inspired Alternatives to Backpropagation Through Time for Learning in Recurrent Neural Nets," *arXiv:1901.09049*, 2019.

[37] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, "A Review of Learning in Biologically Plausible Spiking Neural Networks," *Neural Networks*, vol. 122, pp. 253–272, 2020.

[38] H. Ghaemi, E. Mirzaei, M. Nouri, and S. R. Kheradpisheh, "BioLCNet: Reward-Modulated Locally Connected Spiking Neural Networks," in *International Online & Onsite Conference on Machine Learning, Optimization, and Data Science*, 2022, pp. 564–578.

[39] Z. Bing, C. Meschede, K. Huang, G. Chen, F. Rohrbein, M. Akl, and A. Knoll, "End to End Learning of Spiking Neural Network Based on R-STDP for a Lane Keeping Vehicle," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 4725–4732.

[40] M. Pfeiffer and T. Pfeil, "Deep Learning with Spiking Neurons: Opportunities and Challenges," *Frontiers in Neuroscience*, vol. 12, 2018, art. no. 774.

[41] A. Sboev, D. Vlasov, R. Rybka, and A. Serenko, "Spiking Neural Network Reinforcement Learning Method Based on Temporal Coding and STDP," *Procedia Computer Science*, vol. 145, pp. 458–463, Jan. 2018.

[42] Z. Bing, Z. Jiang, L. Cheng, C. Cai, K. Huang, and A. C. Knoll, "End to End Learning of a Multi-Layered SNN Based on R-STDP for a Target Tracking Snake-Like Robot," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 9645–9651.

[43] H. Asgari, B. M.-N. Maybodi, M. Payvand, and M. R. Azghadi, "Low-Energy and Fast Spiking Neural Network For Context-Dependent Learning on FPGA," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 11, pp. 2697–2701, Nov. 2020.

[44] T. Wunderlich, A. F. Kungl, E. Müller, A. Hartel, Y. Stradmann, S. A. Aamir, A. Grübl, A. Heimbrecht, K. Schreiber, D. Stöckel, C. Pehle, S. Billaudelle, G. Kiene, C. Mauch, J. Schemmel, K. Meier, and M. A. Petrovici, "Demonstrating Advantages of Neuromorphic Computation: A Pilot Study," *Frontiers in Neuroscience*, vol. 13, Mar. 2019, art. no. 260.

[45] N. Frémaux, H. Sprekeler, and W. Gerstner, "Reinforcement Learning Using a Continuous Time Actor-Critic Framework with Spiking Neurons," *PLoS Computational Biology*, vol. 9, no. 4, 2013, art. no. e1003024.

[46] S. Chung and R. Kozma, "Reinforcement Learning with Feedback-modulated TD-STDP," *arXiv:2008.13044*, Aug. 2020.

[47] S. Guo, "State and Action Abstraction in Reinforcement Learning," Ph.D. dissertation, Tsinghua University, May 2021.

[48] A. Borji and L. Itti, "State-of-the-Art in Visual Attention Modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 185–207, Jan. 2013.

[49] Z. Yang, S. Mondal, S. Ahn, G. Zelinsky, M. Hoai, and D. Samaras, "Predicting Human Attention using Computational Attention," *arXiv:2303.09383*, Apr. 2023.

[50] M. Zhang, J. Feng, K. T. Ma, J. H. Lim, Q. Zhao, and G. Kreiman, "Finding Any Waldo with Zero-Shot Invariant and Efficient Visual Search," *Nature Communications*, vol. 9, no. 1, Sep. 2018, art. no. 3730.

[51] S. K. Gupta, M. Zhang, C.-C. Wu, J. M. Wolfe, and G. Kreiman, "Visual Search Asymmetry: Deep Nets and Humans Share Similar Inherent Biases," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 6946–6959.

[52] S. Mondal, Z. Yang, S. Ahn, D. Samaras, G. Zelinsky, and M. Hoai, "Gazeformer: Scalable, Effective and Fast Prediction of Goal-Directed Human Attention," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2023, pp. 1441–1450.

[53] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang, "Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2693–2708, Nov. 2019.

[54] X. Chen, M. Jiang, and Q. Zhao, "Predicting Human Scanpaths in Visual Question Answering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 876–10 885.

[55] Z. Yang, L. Huang, Y. Chen, Z. Wei, S. Ahn, G. J. Zelinsky, D. Samaras, and M. Hoai, "Predicting Goal-Directed Human Attention Using Inverse Reinforcement Learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 190–199.

[56] X. Chen, L. Yao, X. Wang, A. Sun, and Q. Z. Sheng, "Generative Adversarial Reward Learning for Generalized Behavior Tendency Inference," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 10, pp. 9878–9889, Oct. 2023.

[57] K. Zhang, G. Tong, and X. Zhang, "Imitating Human Selective Attention Using Dual Policy Network for Scanpath Prediction," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr. 2025, pp. 1–5.

[58] S. Mathe and C. Sminchisescu, "Action from Still Image Dataset and Inverse Optimal Control to Learn Task Specific Visual Scanpaths," in *Advances in Neural Information Processing Systems*, vol. 26. Curran Associates, Inc., 2013.

[59] Z. Yang, S. Mondal, S. Ahn, G. Zelinsky, M. Hoai, and D. Samaras, "Target-Absent Human Attention," in *Computer Vision – ECCV 2022*. Springer Nature Switzerland, 2022, pp. 52–68.

[60] S. Chakraborty, Z. Wei, C. Kelton, S. Ahn, A. Balasubramanian, G. J. Zelinsky, and D. Samaras, "Predicting Visual Attention in Graphic Design Documents," *IEEE Transactions on Multimedia*, vol. 25, pp. 4478–4493, 2022.

[61] Y. Zhou, D. Han, and Y. Yu, "Energy-Efficient Visual Search by Eye Movement and Low-Latency Spiking Neural Network," *arXiv:2310.06578*, Oct. 2023.

[62] Z. Yu, S. Guo, F. Deng, Q. Yan, K. Huang, J. K. Liu, and F. Chen, "Emergent Inference of Hidden Markov Models in Spiking Neural Networks Through Winner-Take-All," *IEEE Transactions on Cybernetics*, vol. 50, no. 3, pp. 1347–1354, Mar. 2020.

[63] S. Guo, Z. Yu, F. Deng, X. Hu, and F. Chen, "Hierarchical Bayesian Inference and Learning in Spiking Neural Networks," *IEEE Transactions on Cybernetics*, vol. 49, no. 1, pp. 133–145, Jan. 2019.

[64] H. Jang, N. Skatchkovsky, and O. Simeone, "VOWEL: A Local Online Learning Rule for Recurrent Networks of Probabilistic Spiking Winner-Take-All Circuits." in *International Conference on Pattern Recognition*, 2020, pp. 4597–4604.

[65] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.

[66] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST Database of Handwritten Digits," 1998, http://yann.lecun.com/exdb/mnist/.

[67] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv:1606.01540*, 2016.

[68] M. Wydmuch, M. Kempka, and W. Jaśkowski, "ViZDoom Competitions: Playing Doom from Pixels," *IEEE Transactions on Games*, vol. 11, no. 3, pp. 248–259, Sep. 2019.

[69] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv:1712.05474*, Dec. 2017.

[70] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi, "RoboTHOR: An Open

Simulation-to-Real Embodied AI Platform," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp. 3161–3171.

[71] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A Versatile and Scalable Robot Simulation Framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326.

[72] C. Li, R. Chen, C. Moutafis, and S. Furber, "Robustness to Noisy Synaptic Weights in Spiking Neural Networks," in *International Joint Conference on Neural Networks*, Jul. 2020, pp. 1–8.

[73] C. A. Manrique Escobar, C. M. Pappalardo, and D. Guida, "A Parametric Study of a Deep Reinforcement Learning Control System Applied to the Swing-Up Problem of the Cart-Pole," *Applied Sciences*, vol. 10, no. 24, Jan. 2020, art. no. 9013.

[74] Z. Yang, S. Guo, Y. Fang, Z. Yu, and J. K. Liu, "Spiking Variational Policy Gradient for Brain Inspired Reinforcement Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 3, pp. 1975–1990, 2024.

[75] R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992.

[76] J. A. Boyan, "Least-Squares Temporal Difference Learning," in *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999, pp. 49–56.

[77] T. J. Walsh, I. Szita, C. Diuk, and M. L. Littman, "Exploring Compact Reinforcement-Learning Representations with Linear Regression," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence.* AUAI Press, Jun. 2009, pp. 591–598.

[78] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 2094–2100.

[79] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust Region Policy Optimization," in *Proceedings of the 32nd International Conference on Machine Learning.* PMLR, 2015, pp. 1889–1897.

[80] S. Zhang and R. S. Sutton, "A Deeper Look at Experience Replay," *arXiv:1712.01275*, Apr. 2018.

[81] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in Deep Reinforcement Learning," *Knowledge-Based Systems*, vol. 214, Feb. 2021, art. no. 106685.

[82] N. Justesen and S. Risi, "Automated Curriculum Learning by Rewarding Temporally Rare Events," in *IEEE Conference on Computational Intelligence and Games*, 2018, pp. 1–8.

[83] D. Ha and J. Schmidhuber, "World Models," *arXiv:1803.10122*, Mar. 2018.

[84] A. Anand, E. Racah, S. Ozair, Y. Bengio, M.-A. Côté, and R. D. Hjelm, "Unsupervised State Representation Learning in Atari," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[85] T. Sadamoto, A. Chakrabortty, and J.-i. Imura, "Fast Online Reinforcement Learning Control Using State-Space Dimensionality Reduction," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 1, pp. 342–353, Mar. 2021.

[86] O. Nachum, H. Tang, X. Lu, S. Gu, H. Lee, and S. Levine, "Why Does Hierarchy (Sometimes) Work So Well in Reinforcement Learning?" *arXiv:1909.10618*, Dec. 2019.

[87] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "Deep Successor Reinforcement Learning," *arXiv:1606.02396*, Jun. 2016.

[88] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *IEEE International Conference on Robotics and Automation*, May 2018, pp. 3803–3810.

[89] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in *Proceedings of the 34th International Conference on Machine Learning.* PMLR, Jul. 2017, pp. 1126–1135.

[90] I. Adamski, R. Adamski, T. Grel, A. Jędrych, K. Kaczmarek, and H. Michalewski, "Distributed Deep Reinforcement Learning: Learn How to Play Atari Games in 21 Minutes," in *International Conference on High Performance Computing.* Springer, 2018, pp. 370–388.

[91] M. A. Issa, H. Chen, J. Wang, and M. Imani, "CyberRL: Brain-Inspired Reinforcement Learning for Efficient Network Intrusion Detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 1, pp. 241–250, Jan. 2025.

[92] S. R. Razavi Rohani, S. Hedayatian, and M. S. Baghshah, "BIMRL: Brain Inspired Meta Reinforcement Learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2022, pp. 9048–9053.

[93] A. Ororbia and A. Mali, "Active Predictive Coding: Brain-Inspired Reinforcement Learning for Sparse Reward Robotic Control Problems," in *IEEE International Conference on Robotics and Automation*, May 2023, pp. 3015–3021.

[94] S. Schmidgall, R. Ziaei, J. Achterberg, L. Kirsch, S. P. Hajiseyedrazi, and J. Eshraghian, "Brain-Inspired Learning in Artificial Neural Networks: A Review," *APL Machine Learning*, vol. 2, no. 2, May 2024, art. no. 021501.

[95] J. Campbell, "Considerations of Biological Plausibility in Deep Learning," *Cornell Undergraduate Research Journal*, vol. 1, no. 1, pp. 4–12, Apr. 2022.

[96] B. Illing, W. Gerstner, and J. Brea, "Biologically Plausible Deep Learning — But How Far Can We Go with Shallow Networks?" *Neural Networks*, vol. 118, pp. 90–101, Oct. 2019.

[97] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep Learning in Spiking Neural Networks," *Neural Networks*, vol. 111, pp. 47–63, Mar. 2019.

[98] Y. Hao, X. Huang, M. Dong, and B. Xu, "A Biologically Plausible Supervised Learning Method for Spiking Neural Networks Using the Symmetric STDP Rule," *Neural Networks*, vol. 121, pp. 387–395, Jan. 2020.

[99] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going Deeper in Spiking Neural Networks: VGG and Residual Architectures," *Frontiers in Neuroscience*, vol. 13, Mar. 2019, art. no. 95.

[100] Q. T. Pham, T. Q. Nguyen, P. C. Hoang, Q. H. Dang, D. M. Nguyen, and H. H. Nguyen, "A Review of SNN Implementation on FPGA," in *International Conference on Multimedia Analysis and Pattern Recognition*, 2021, pp. 1–6.

[101] A. L. Hodgkin and A. F. Huxley, "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[102] E. M. Izhikevich, "Simple Model of Spiking Neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[103] ——, "Which Model to Use for Cortical Spiking Neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[104] R. Jolivet, T. J., and W. Gerstner, "The Spike Response Model: A Framework to Predict Neuronal Spike Trains," in *Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP*. Springer, 2003, pp. 846–853.

[105] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass, "Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity," *PLoS Computational Biology*, vol. 9, no. 4, 2013, art. no. e1003037.

[106] N. S. Giraldo, S. Isaza, and R. A. Velásquez, "Sailboat Navigation Control System Based on Spiking Neural Networks," *Control Theory and Technology*, vol. 21, no. 4, pp. 489–504, Aug. 2023.

[107] R. V. Florian, "Reinforcement Learning Through Modulation of Spike-Timing-Dependent Synaptic Plasticity," *Neural Computation*, vol. 19, no. 6, pp. 1468–1502, Jun. 2007.

[108] Y. C. Yoon, "LIF and Simplified SRM Neurons Encode Signals Into Spikes via a Form of Asynchronous Pulse Sigma–Delta Modulation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 5, pp. 1192–1205, May 2017.

[109] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition.* Cambridge University Press, 2014.

[110] M. Ambard and S. Rotter, "Support Vector Machines for Spike Pattern Classification with a Leaky Integrate-and-Fire Neuron," *Frontiers in Computational Neuroscience*, vol. 6, Nov. 2012, art. no. 78.

[111] I. Carannante, Y. Johansson, G. Silberberg, and J. Hellgren Kotaleski, "Data-Driven Model of Postsynaptic Currents Mediated by NMDA or AMPA Receptors in Striatal Neurons," *Frontiers in Computational Neuroscience*, vol. 16, May 2022, art. no. 806086.

[112] G. Tang, N. Kumar, and K. P. Michmizos, "Reinforcement co-Learning of Deep and Spiking Neural Networks for Energy-Efficient Mapless Navigation with Neuromorphic Hardware," in *IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, Oct. 2020, pp. 6090–6097.

[113] G. Tang, N. Kumar, R. Yoo, and K. Michmizos, "Deep Reinforcement Learning with Population-Coded Spiking Neural Network for Continuous Control," in *Conference on Robot Learning.* PMLR, Oct. 2021, pp. 2016–2029.

[114] G. Liu, W. Deng, X. Xie, L. Huang, and H. Tang, "Human-Level Control Through Directly Trained Deep Spiking Q-Networks," *IEEE Transactions on Cybernetics*, vol. 53, no. 11, pp. 7187–7198, Nov. 2023.

[115] D. Zhang, T. Zhang, S. Jia, Q. Wang, and B. Xu, "Recent Advances and New Frontiers in Spiking Neural Networks," in *International Joint Conference on Artificial Intelligence*, 2022, pp. 5670–5677.

[116] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep Residual Learning in Spiking Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 21 056–21 069.

[117] Z. Zheng, Y. Huang, Y. Yu, Z. Zhu, J. Tang, Z. Yu, and Y. Jin, "SpiLiFormer: Enhancing Spiking Transformers with Lateral Inhibition," *arXiv:2503.15986*, Mar. 2025.

[118] W. Maass, T. Natschläger, and H. Markram, "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[119] S. Slade and L. Zhang, "Topological Evolution of Spiking Neural Networks," in *International Joint Conference on Neural Networks*, Jul. 2018, pp. 1–9.

[120] S. Dora, S. Sundaram, and N. Sundararajan, "A Two Stage Learning Algorithm for a Growing-Pruning Spiking Neural Network for Pattern Classification Problems," in *International Joint Conference on Neural Networks*, Jul. 2015, pp. 1–7.

[121] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," in *International Conference on Learning Representations*, 2016.

[122] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy Networks for Exploration," in *International Conference on Learning Representations*, 2018.

[123] Y. Cao, Y. Chen, and D. Khosla, "Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, May 2015.

[124] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of Artificial Recurrent Neural Networks to Spiking Neural Networks for Low-Power Neuromorphic Hardware," in *IEEE International Conference on Rebooting Computing.* IEEE, Oct. 2016, pp. 1–8.

[125] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification," *Frontiers in Neuroscience*, vol. 11, Dec. 2017, art. no. 682.

[126] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, Nov. 2019.

[127] Y. Li, Y. Guo, S. Zhang, S. Deng, Y. Hai, and S. Gu, "Differentiable Spike: Rethinking Gradient-Descent for Training Spiking Neural Networks," in *Annual Conference on Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 23 426–23 439.

[128] C. Shi, T. Wang, J. He, J. Zhang, L. Liu, and N. Wu, "DeepTempo: A Hardware-Friendly Direct Feedback Alignment Multi-Layer Tempotron Learning Rule for Deep Spiking Neural Networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 5, pp. 1581–1585, May 2021.

[129] A. Vigneron and J. Martinet, "A Critical Survey of STDP in Spiking Neural Networks for Pattern Recognition," in *International Joint Conference on Neural Networks*, Jul. 2020, pp. 1–9.

[130] J. N. J. Reynolds, B. I. Hyland, and J. R. Wickens, "A Cellular Mechanism of Reward-Related Learning," *Nature*, vol. 413, no. 6851, pp. 67–70, Sep. 2001.

[131] D. E. Shulz, R. Sosnik, V. Ego, S. Haidarliu, and E. Ahissar, "A Neuronal Analogue of State-Dependent Learning," *Nature*, vol. 403, no. 6769, pp. 549–553, Feb. 2000.

[132] A. Tavanaei and A. Maida, "BP-STDP: Approximating Backpropagation Using Spike Timing Dependent Plasticity," *Neurocomputing*, vol. 330, pp. 39–47, 2019.

[133] C. Lee, P. Panda, G. Srinivasan, and K. Roy, “Training Deep Spiking Convolutional Neural Networks With STDP-Based Unsupervised Pre-training Followed by Supervised Fine-Tuning,” *Frontiers in Neuroscience*, vol. 12, Aug. 2018, art. no. 435.

[134] D. Chen, P. Peng, T. Huang, and Y. Tian, “Deep Reinforcement Learning with Spiking Q-learning,” *arXiv:2201.09754*, 2022.

[135] G. O’Shea and M. Komeili, “SuperVision: Self-Supervised Super-Resolution for Appearance-Based Gaze Estimation,” in *NeuRIPS 2023 Workshop on Gaze Meets ML*, Oct. 2023.

[136] C. Kuang, J. O. Kephart, and Q. Ji, “Interaction-aware Dynamic 3D Gaze Estimation in Videos,” in *NeuRIPS 2023 Workshop on Gaze Meets ML*, Oct. 2023.

[137] A. Nakazawa and C. Nitschke, “Point of Gaze Estimation through Corneal Surface Reflection in an Active Illumination Environment,” in *Computer Vision – ECCV 2012*. Springer, 2012, pp. 159–172.

[138] Y. Tao and M.-L. Shyu, “SP-ASDNet: CNN-LSTM Based ASD Classification Model using Observer ScanPaths,” in *IEEE International Conference on Multimedia & Expo Workshops*, Jul. 2019, pp. 641–646.

[139] G. J. Zelinsky, Z. Yang, L. Huang, Y. Chen, S. Ahn, Z. Wei, H. Adeli, D. Samaras, and M. Hoai, “Benchmarking Gaze Prediction for Categorical Visual Search,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019, pp. 4321–4329.

[140] Y. Chen, Z. Yang, S. Ahn, D. Samaras, M. Hoai, and G. Zelinsky, “COCO-Search18 Fixation Dataset for Predicting Goal-Directed Attention Control,” *Scientific Reports*, vol. 11, no. 1, Apr. 2021, art. no. 8776.

[141] S. Rashidi, K. Ehinger, A. Turpin, and L. Kulik, “Optimal Visual Search Based on a Model of Target Detectability in Natural Images,” in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 9288–9299.

[142] F. Travi, G. Ruarte, G. Bujia, and J. E. Kamienkowski, “ViSioNS: Visual Search in Natural Scenes Benchmark,” in *Advances in Neural Information Processing Systems*, vol. 35. Curran Associates, Inc., Dec. 2022, pp. 11 987–12 000.

[143] H. Griffith, D. Lohr, E. Abdulin, and O. Komogortsev, “GazeBase, a Large-Scale, Multi-Stimulus, Longitudinal Eye Movement Dataset,” *Scientific Data*, vol. 8, no. 1, Jul. 2021, art. no. 184.

[144] K. Melnyk, L. Friedman, D. Katrychuk, and O. Komogortsev, “Per-Subject Oculomotor Plant Mathematical Models and the Reliability of Their Parameters,” *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 7, no. 2, pp. 1–20, May 2024, art. no. 24.

[145] S. Baee, E. Pakdamanian, I. Kim, L. Feng, V. Ordonez, and L. Barnes, “MEDIRL: Predicting the Visual Attention of Drivers via Maximum Entropy Deep Inverse Reinforcement Learning,” in *IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 178–13 188.

[146] Z. Lian, T. Xu, Z. Yuan, J. Li, N. Thakor, and H. Wang, “Driving Fatigue Detection Based on Hybrid Electroencephalography and Eye Tracking,” *IEEE Journal of Biomedical and Health Informatics*, vol. 28, no. 11, pp. 6568–6580, Nov. 2024.

[147] T. L. Botch, B. D. Garcia, Y. B. Choi, N. Feffer, and C. E. Robertson, “Active Visual Search in Naturalistic Environments Reflects Individual Differences in Classic Visual Search Performance,” *Scientific Reports*, vol. 13, no. 1, Jan. 2023, art. no. 631.

[148] C. S. Burlingham, N. Sendhilnathan, O. Komogortsev, T. S. Murdison, and M. J. Proulx, “Motor “Laziness” Constrains Fixation Selection in Real-World Tasks,” *Proceedings of the National Academy of Sciences*, vol. 121, no. 12, Mar. 2024, art. no. e2302239121.

[149] S. Losorelli, J. K. Chang, K. W. Chang, S. P. Most, and M. T. Truong, “Gaze Patterns of Normal and Microtia Ears Pre- and Post-Reconstruction,” *The Laryngoscope*, vol. 134, no. 7, pp. 3136–3142, Feb. 2024.

[150] S. Eraslan, Y. Yesilada, V. Yaneva, and S. Harper, “Autism Detection Based on Eye Movement Sequences on the Web: A Scanpath Trend Analysis Approach,” in *Proceedings of the 17th International Web for All Conference*. Association for Computing Machinery, Apr. 2020, pp. 1–10, art. no. 11.

[151] A. B. Dris, A. Alsalman, A. Al-Wabil, and M. Aldosari, “Intelligent Gaze-Based Screening System for Autism,” in *International Conference on Computer Applications & Information Security*, May 2019, pp. 1–5.

[152] Y. Chen, Z. Yang, S. Chakraborty, S. Mondal, S. Ahn, D. Samaras, M. Hoai, and G. Zelinsky, “Characterizing Target-Absent Human Attention,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5031–5040.

[153] M. Assens, X. Giro-i-Nieto, K. McGuinness, and N. E. O’Connor, “PathGAN: Visual Scanpath Prediction with Generative Adversarial Networks,” in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 1–18.

[154] W. Sun, Z. Chen, and F. Wu, “Visual Scanpath Prediction Using IOR-ROI Recurrent Mixture Density Network,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 6, pp. 2101–2118, Jun. 2021.

[155] C. Xia, F. Qi, and G. Shi, “An Iterative Representation Learning Framework to Predict the Sequence of Eye Fixations,” in *IEEE International Conference on Multimedia and Expo*, Jul. 2017, pp. 1530–1535.

[156] J. Najemnik and W. S. Geisler, "Optimal Eye Movement Strategies in Visual Search," *Nature*, vol. 434, no. 7031, pp. 387–391, Mar. 2005.

[157] G. Bujia, M. Sclar, S. Vita, G. Solovey, and J. E. Kamienkowski, "Modeling Human Visual Search in Natural Scenes: A Combined Bayesian Searcher and Saliency Map Approach," *Frontiers in Systems Neuroscience*, vol. 16, 2022, art. no. 882315.

[158] M. Kümmerer, M. Bethge, and T. S. A. Wallis, "DeepGaze III: Modeling Free-Viewing Human Scanpaths with Deep Learning," *Journal of Vision*, vol. 22, no. 5, Apr. 2022, art. no. 7.

[159] G. J. Zelinsky and H. Adeli, "Learning to Attend in a Brain-Inspired Deep Neural Network," *Journal of Vision*, vol. 19, no. 10, Sep. 2019, art. no. 282d.

[160] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," in *AAAI Conference on Artificial Intelligence*, 2008, pp. 1433–1438.

[161] C. Finn, S. Levine, and P. Abbeel, "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization," in *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, Jun. 2016, pp. 49–58.

[162] A. Y. Ng and S. Russell, "Algorithms for Inverse Reinforcement Learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 663–670.

[163] D. Ramachandran and E. Amir, "Bayesian Inverse Reinforcement Learning," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., Jan. 2007, pp. 2586–2591.

[164] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[165] D. Garg, S. Chakraborty, C. Cundy, J. Song, and S. Ermon, "IQ-Learn: Inverse Soft-Q Learning for Imitation," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 4028–4039.

[166] Z. Bylinskii, T. Judd, A. Oliva, A. Torralba, and F. Durand, "What Do Different Evaluation Metrics Tell Us About Saliency Models?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 3, pp. 740–757, Mar. 2019.

[167] K. H. Ruddock, D. S. Wooding, and S. Mannan, "Automatic Control of Saccadic Eye Movements Made in Visual Inspection of Briefly Presented 2-D Images," *Spatial Vision*, vol. 9, no. 3, pp. 363–386, Jan. 1995.

[168] R. Dewhurst, M. Nyström, H. Jarodzka, T. Foulsham, R. Johansson, and K. Holmqvist, "It Depends on How You Look at It: Scanpath Comparison in Multiple Dimensions with MultiMatch, a Vector-Based Approach," *Behavior Research Methods*, vol. 44, no. 4, pp. 1079–1100, Dec. 2012.

[169] V. I. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions, and Reversals,” in *Soviet Physics Doklady*, vol. 10. Soviet Union, 1966, pp. 707–710.

[170] F. Cristino, S. Mathôt, J. Theeuwes, and I. D. Gilchrist, “ScanMatch: A Novel Method for Comparing Fixation Sequences,” *Behavior Research Methods*, vol. 42, no. 3, pp. 692–700, Aug. 2010.

[171] V. Narayanan and M. Likhachev, “PERCH: Perception via search for multi-object recognition and localization,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 5052–5059.

[172] Z. Huang, Y. Zhou, J. Zhu, and C. Gou, “Driver Scanpath Prediction Based On Inverse Reinforcement Learning,” in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2024, pp. 8306–8310.

[173] M. Jiang and Q. Zhao, “Learning Visual Attention to Identify People with Autism Spectrum Disorder,” in *IEEE International Conference on Computer Vision*, Oct. 2017, pp. 3287–3296.

[174] S. Wang, M. Jiang, X. M. Duchesne, E. A. Laugeson, D. P. Kennedy, R. Adolphs, and Q. Zhao, “Atypical Visual Saliency in Autism Spectrum Disorder Quantified through Model-Based Eye Tracking,” *Neuron*, vol. 88, no. 3, pp. 604–616, Nov. 2015.

[175] M. L. Spezio, R. Adolphs, R. S. E. Hurley, and J. Piven, “Analysis of Face Gaze in Autism Using “Bubbles”,” *Neuropsychologia*, vol. 45, no. 1, pp. 144–151, Jan. 2007.

[176] P.-H. Tseng, I. G. M. Cameron, G. Pari, J. N. Reynolds, D. P. Munoz, and L. Itti, “High-Throughput Classification of Clinical Populations from Natural Viewing Eye Movements,” *Journal of Neurology*, vol. 260, no. 1, pp. 275–284, Jan. 2013.

[177] W. Liu, M. Li, and L. Yi, “Identifying Children with Autism Spectrum Disorder Based on Their Face Processing Abnormality: A Machine Learning Framework,” *Autism Research*, vol. 9, no. 8, pp. 888–898, 2016.

[178] H. Duan, G. Zhai, X. Min, Y. Fang, Z. Che, X. Yang, C. Zhi, H. Yang, and N. Liu, “Learning to Predict where the Children with ASD Look,” in *IEEE International Conference on Image Processing*, Oct. 2018, pp. 704–708.

[179] W. Wei, Z. Liu, L. Huang, A. Nebout, and O. Le Meur, “Saliency Prediction via Multi-Level Features and Deep Supervision for Children with Autism Spectrum Disorder,” in *IEEE International Conference on Multimedia & Expo Workshops*, Jul. 2019, pp. 621–624.

[180] N. Heess, D. Silver, and Y. W. Teh, “Actor-Critic Reinforcement Learning with Energy-Based Policies,” in *European Workshop on Reinforcement Learning*, vol. 24, 2012, pp. 43–58.

[181] M. J. Wainwright and M. I. Jordan, “Graphical Models, Exponential Families, and Variational Inference,” *Foundations and Trends® in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.

[182] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[183] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, "ViZDoom: A Doom-Based AI Research Platform for Visual Reinforcement Learning," in *IEEE Conference on Computational Intelligence and Games*, 2016, pp. 1–8.

[184] S. James, M. Freese, and A. J. Davison, "PyRep: Bringing V-REP to Deep Robot Learning," *arXiv:1906.11176*, 2019.

[185] Â. G. Lovatto, T. P. Bueno, and L. N. Barros, "Analyzing the Effect of Stochastic Transitions in Policy Gradients in Deep Reinforcement Learning," in *Brazilian Conference on Intelligent Systems*, Oct. 2019, pp. 413–418.

[186] R. Özalp, N. K. Varol, B. Taşci, and A. Uçar, "A Review of Deep Reinforcement Learning Algorithms and Comparative Results on Inverted Pendulum System," in *Machine Learning Paradigms: Advances in Deep Learning-based Technological Applications*. Springer International Publishing, 2020, pp. 237–256.

[187] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[188] W. Fang, Y. Chen, J. Ding, Z. Yu, T. Masquelier, D. Chen, L. Huang, H. Zhou, G. Li, and Y. Tian, "SpikingJelly: An Open-Source Machine Learning Infrastructure Platform for Spike-Based Intelligence," *Science Advances*, vol. 9, no. 40, Oct. 2023, art. no. eadi1480.

[189] F. Zenke and S. Ganguli, "SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks," *Neural Computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[190] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training Spiking Neural Networks Using Lessons From Deep Learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, Sep. 2023.

[191] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Bio-Inspired Digit Recognition Using Reward-Modulated Spike-Timing-Dependent Plasticity in Deep Convolutional Networks," *Pattern Recognition*, vol. 94, pp. 87–95, Oct. 2019.

[192] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, and T. Masquelier, "SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks With at Most One Spike per Neuron," *Frontiers in Neuroscience*, vol. 13, Jul. 2019, art. no. 625.

[193] S. Aenugu, A. Sharma, S. Yelamarthy, H. Hazan, Philip.S.Thomas, and R. Kozma, “Reinforcement Learning with a Network of Spiking Agents,” in *Real Neurons & Hidden Units: Future Directions at the Intersection of Neuroscience and Artificial Intelligence @ NeurIPS 2019*, Jul. 2022, pp. 1–5.

[194] Z. Yang, S. Guo, Y. Fang, and J. Liu, “Biologically Plausible Variational Policy Gradient with Spiking Recurrent Winner-Take-All Networks,” in *British Machine Vision Conference*, 2022, art. no. 358.

[195] P. R. Winters, “Forecasting Sales by Exponentially Weighted Moving Averages,” *Management Science*, vol. 6, no. 3, pp. 324–342, 1960.

[196] S. Sharmin, P. Panda, S. S. Sarwar, C. Lee, W. Ponghiran, and K. Roy, “A Comprehensive Analysis on Adversarial Robustness of Spiking Neural Networks,” in *International Joint Conference on Neural Networks*, 2019, pp. 1–8.

[197] N. Zheng and P. Mazumder, “Learning in Memristor Crossbar-Based Spiking Neural Networks Through Modulation of Weight-Dependent Spike-Timing-Dependent Plasticity,” *IEEE Transactions on Nanotechnology*, vol. 17, no. 3, pp. 520–532, May 2018.

[198] A. J. Snoswell, S. P. N. Singh, and N. Ye, “Revisiting Maximum Entropy Inverse Reinforcement Learning: New Perspectives and Algorithms,” in *IEEE Symposium Series on Computational Intelligence*, Dec. 2020, pp. 241–249.

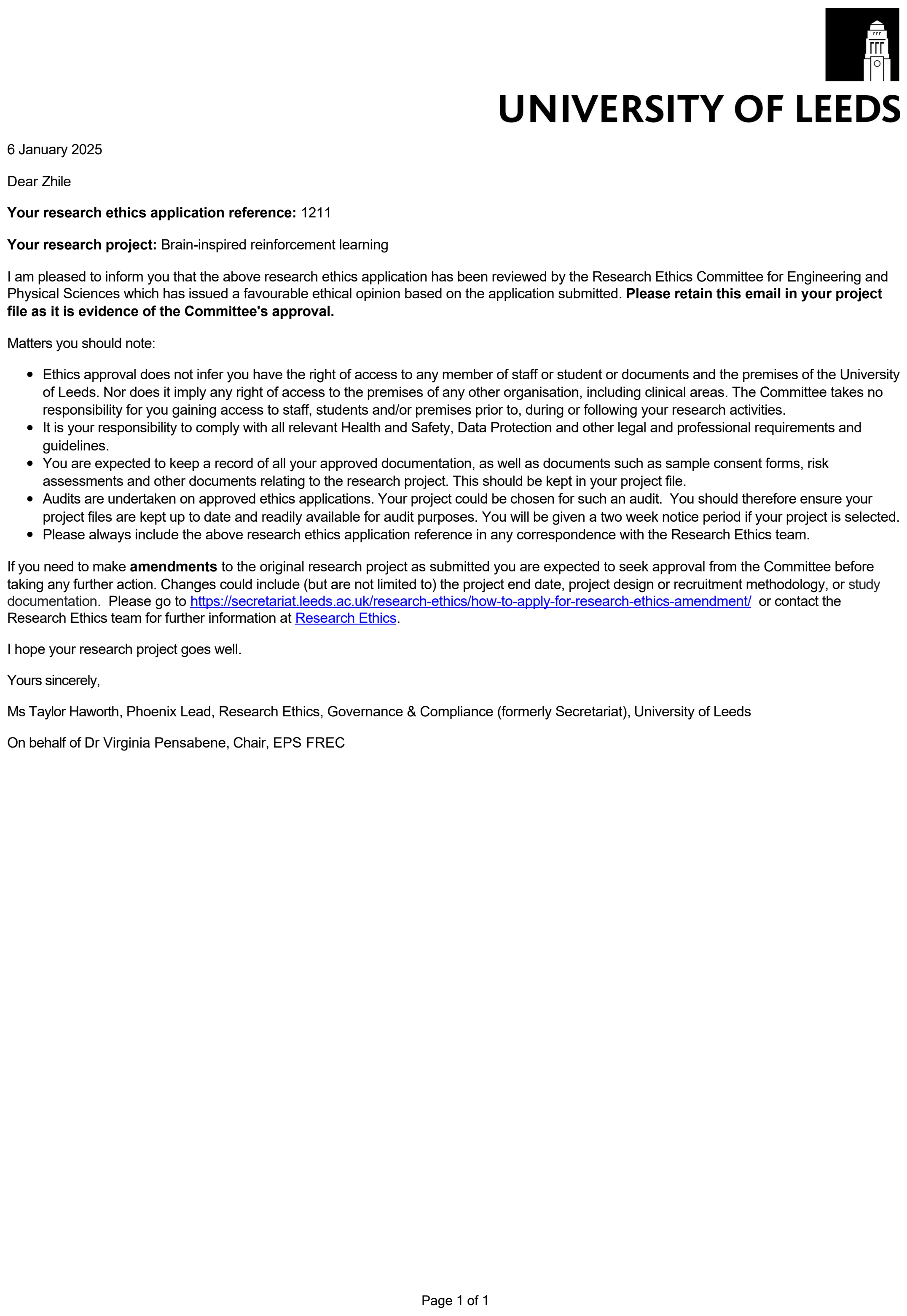
[199] B. H. Giwa, “Discount Factor Estimation in Inverse Reinforcement Learning,” Ph.D. dissertation, University of Toronto (Canada), 2022.

[200] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum Entropy Deep Inverse Reinforcement Learning,” *arXiv:1507.04888*, Mar. 2016.

[201] W. Wang, C. Chen, Y. Wang, T. Jiang, F. Fang, and Y. Yao, “Simulating Human Saccadic Scanpaths on Natural Images,” in *Conference on Computer Vision and Pattern Recognition*, Jun. 2011, pp. 441–448.

Appendix A

Ethics Review



UNIVERSITY OF LEEDS

6 January 2025

Dear Zhile

Your research ethics application reference: 1211

Your research project: Brain-inspired reinforcement learning

I am pleased to inform you that the above research ethics application has been reviewed by the Research Ethics Committee for Engineering and Physical Sciences which has issued a favourable ethical opinion based on the application submitted. **Please retain this email in your project file as it is evidence of the Committee's approval.**

Matters you should note:

- Ethics approval does not infer you have the right of access to any member of staff or student or documents and the premises of the University of Leeds. Nor does it imply any right of access to the premises of any other organisation, including clinical areas. The Committee takes no responsibility for you gaining access to staff, students and/or premises prior to, during or following your research activities.
- It is your responsibility to comply with all relevant Health and Safety, Data Protection and other legal and professional requirements and guidelines.
- You are expected to keep a record of all your approved documentation, as well as documents such as sample consent forms, risk assessments and other documents relating to the research project. This should be kept in your project file.
- Audits are undertaken on approved ethics applications. Your project could be chosen for such an audit. You should therefore ensure your project files are kept up to date and readily available for audit purposes. You will be given a two week notice period if your project is selected.
- Please always include the above research ethics application reference in any correspondence with the Research Ethics team.

If you need to make **amendments** to the original research project as submitted you are expected to seek approval from the Committee before taking any further action. Changes could include (but are not limited to) the project end date, project design or recruitment methodology, or study documentation. Please go to https://secretariat.leeds.ac.uk/research-ethics/how-to-apply-for-research-ethics-amendment/ or contact the Research Ethics team for further information at Research Ethics.

I hope your research project goes well.

Yours sincerely,

Ms Taylor Haworth, Phoenix Lead, Research Ethics, Governance & Compliance (formerly Secretariat), University of Leeds

On behalf of Dr Virginia Pensabene, Chair, EPS FREC

Page 1 of 1