UNIVERSITY OF CALIFORNIA,
IRVINE

Understanding the Brain using Machine Learning and Enhancing Machine Learning with Neuroscience

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Cognitive Sciences

by

Jinwei Xing

Dissertation Committee:
Professor Jeffrey L. Krichmar, Chair
Assistant Professor Aaron Bornstein
Associate Professor Sameer Singh

2023

TABLE OF CONTENTS

List of Figures

List of Tables

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

Foremost, I would like to thank my advisor Professor Jeffrey L. Krichmar for his endless guidance, support, inspiration, and patience during my PhD journey. Meeting and working with my advisor is one of the luckiest things in my life. I also sincerely thank Professor Emre Neftci for his constructive mentoring and kind support. His sharing and feedback are always inspiring. I would also like to thank Professor Aaron Bornstein, Sameer Singh and Roy Fox for serving on my committees and providing feedback to strengthen my work.

I also owe my thanks to the following people and/or organizations.

The Cognitive Anteater Robotics Laboratory (CARL). Tiffany Hwu, Hirak Kashyap, Xinyun Zou, Kexin Chen, Ting-Shuo Chou, Tim Lui, Nicholas Alonso, Harrison Espino, Robert K Bain, Seyed Amirhosein Mohaddesi, John Shepanski and Lars Niedermeier for sharing their expert viewpoints and collaborating on diverse projects.

The Neuromorphic Machine Intelligence Lab (NMI). Takashi Nagata, Dan Barsever, Kenneth Stewart and Yue Yin for sharing their interesting viewpoints and thoughtful discussions.

The funding sources that supported my Ph.D. studies: the National Science Foundation (NSF), the Defense Advanced Research Projects Agency (DARPA), and the UCI School of Social Sciences.

Trina Norden-Krichmar for her kind help in English speaking guidance and practice.

My family for supporting my dreams and for being always there with me physically or remotely.

My wife and best friend, Jie Zhu, for always being by my side, encouraging me, and sharing the journey with love and care. I am truly blessed to have you in my life.

VITA

Jinwei Xing

EDUCATION

Doctor of Philosophy in Cognitive Sciences **2023**
University of California, Irvine *Irvine, California*

Bachelor of Science in Computer Science **2017**
Sichuan University *Chengdu, Sichuan*

WORK AND RESEARCH EXPERIENCE

Graduate Student Researcher **2018-2023**
University of California, Irvine *Irvine, California*

Software Engineer Intern **2022**
Google *Sunnyvale, California*

Research Scientist Intern **2021**
Amazon *Irvine, California*

Machine Learning Engineer **2017-2018**
Lieluobo *Shanghai, China*

TEACHING EXPERIENCE

Teaching Assistant **S'19, F'19, W'20, S'20, W'22**
University of California, Irvine *Irvine, California*

PUBLICATIONS

Xing, J., Nagata, T., Zou, X., Neftci, E. and Krichmar, J.L., 2023. Achieving efficient interpretability of reinforcement learning via policy distillation and selective input gradient regularization. Neural Networks, 161, pp.228-241.

Xing, J., Chrastil, E.R., Nitz, D.A. and Krichmar, J.L., 2022. Linking global top-down views to first-person views in the brain. Proceedings of the National Academy of Sciences, 119(45), p.e2202024119.

Xing, J., Zou, X., Pilly, P.K., Ketz, N.A. and Krichmar, J.L., 2022, September. Adapting to Environment Changes Through Neuromodulation of Reinforcement Learning. In From Animals to Animats 16: 16th International Conference on Simulation of Adaptive Behavior, SAB 2022, Cergy-Pontoise, France, September 20–23, 2022, Proceedings (pp. 115-126).

Cham: Springer International Publishing.

Nagata, T., Xing, J., Kumazawa, T. and Neftci, E., 2022, July. Uncertainty Aware Model Integration on Reinforcement Learning. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-7). IEEE.

Niedermeier, L., Chen, K., Xing, J., Das, A., Kopsick, J., Scott, E., Sutton, N., Weber, K., Dutt, N. and Krichmar, J.L., 2022, July. CARLsim 6: An Open Source Library for Large-Scale, Biologically Detailed Spiking Neural Network Simulation. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-10). IEEE.

Kopsick, J.D., Tecuatl, C., Moradi, K., Attili, S.M., Kashyap, H.J., Xing, J., Chen, K., Krichmar, J.L. and Ascoli, G.A., 2022. Robust resting-state dynamics in a large-scale spiking neural network model of area ca3 in the mouse hippocampus. Cognitive Computation, pp.1-21.

Xing, J., Nagata, T., Chen, K., Zou, X., Neftci, E. and Krichmar, J.L., 2021, May. Domain adaptation in reinforcement learning via latent unified state representation. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 12, pp. 10452-10459).

Chen, K., Hwu, T., Kashyap, H.J., Krichmar, J.L., Stewart, K., Xing, J. and Zou, X., 2020. Neurorobots as a means toward neuroethology and explainable AI. Frontiers in Neurorobotics, 14, p.570308.

Xing, J., Zou, X. and Krichmar, J.L., 2020, July. Neuromodulated patience for robot and self-driving vehicle navigation. In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

Chou, T.S., Kashyap, H.J., Xing, J., Listopad, S., Rounds, E.L., Beyeler, M., Dutt, N. and Krichmar, J.L., 2018, July. CARLsim 4: An open source library for large scale, biologically detailed spiking neural network simulation using heterogeneous clusters. In 2018 International joint conference on neural networks (IJCNN) (pp. 1-8). IEEE.

Xu, Y., Tang, H., Xing, J. and Li, H., 2017, November. Spike trains encoding and threshold rescaling method for deep spiking neural networks. In 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1-6). IEEE.

ABSTRACT OF THE DISSERTATION

Understanding the Brain using Machine Learning and Enhancing Machine Learning with Neuroscience

By

Jinwei Xing

Doctor of Philosophy in Cognitive Sciences

University of California, Irvine, 2023

Professor Jeffrey L. Krichmar, Chair

In recent years, machine learning and neuroscience are increasingly intertwined. On the one hand, machine learning could benefit from the insights and inspiration provided by the discoveries in neuroscience, as well as the integration of biologically-inspired components. On the other hand, machine learning techniques can be used to enhance our knowledge of the brain and its functions. In this dissertation, we demonstrate how they could benefit from each other, with an emphasis on dynamic environments. We first introduce how machine learning could benefit from neuroscience. We start with two projects that integrate neuromodulatory systems into machine learning systems to handle dynamic environment changes. In the first project, we used a serotonergic (5-HT) neuromodulatory system to control the patience level of a mobile robot navigating in outdoor environments, resulting in flexible behaviors not typically found in traditional navigation solutions. The second project introduced a reinforcement learning solution augmented with noradrenergic (NE) and cholinergic (ACh) neuromodulation, enabling the agent to quickly adapt to dynamic environment changes. Besides the utilization of neuromodulatory systems, in the third project, we proposed a method of latent unified state representation (LUSR) to improve the domain adaptation performance of reinforcement learning methods by addressing the adaptation problem from the pixel domain to a latent space, inspired by the latent representation in brain. The

fourth project introduced a method of policy distillation with selective input gradient regularization, inspired by memory consolidation, to achieve computation efficiency and high interpretability in explainable reinforcement learning. Finally, the dissertation discusses how machine learning can contribute to the field of neuroscience. The last project studied the transformation between the first-person view and global view, which utilized the machine learning technique of variational autoencoder (VAE) to enhance our understanding of how the brain conducts view transformation in a 3D environment. In summary, this dissertation demonstrated the mutually beneficial relationship between the fields of machine learning and neuroscience, highlighting how each field can help the other to achieve advancements in theory and practice.

Chapter 1

Introduction

The ability to interact with the environment and make decisions based on those interactions is a key aspect of both artificial and biological intelligence. In recent years, research in machine learning has made significant advances in agent-based reinforcement learning, utilizing deep neural networks. However, most of these achievements have been in static environments, where the environment dynamics and reward feedback remain fixed. In reality, the environment can be dynamic and undergo continuous changes, presenting a significant challenge to handling such scenarios.

Decision making under uncertainty and quick adaptation to environment changes is a crucial ability for human and animals to survive in the environment. To accomplish this, the brain employs a variety of processes and mechanisms such as perception, prediction, attention, and learning and memory. These mechanisms and processes could provide inspirations for researchers to augment machine learning and reinforcement learning with brain-inspired components and algorithms, which may assist in addressing the challenge of handling dynamic environments.

Besides providing biological inspirations for machine learning, neuroscience could also benefit

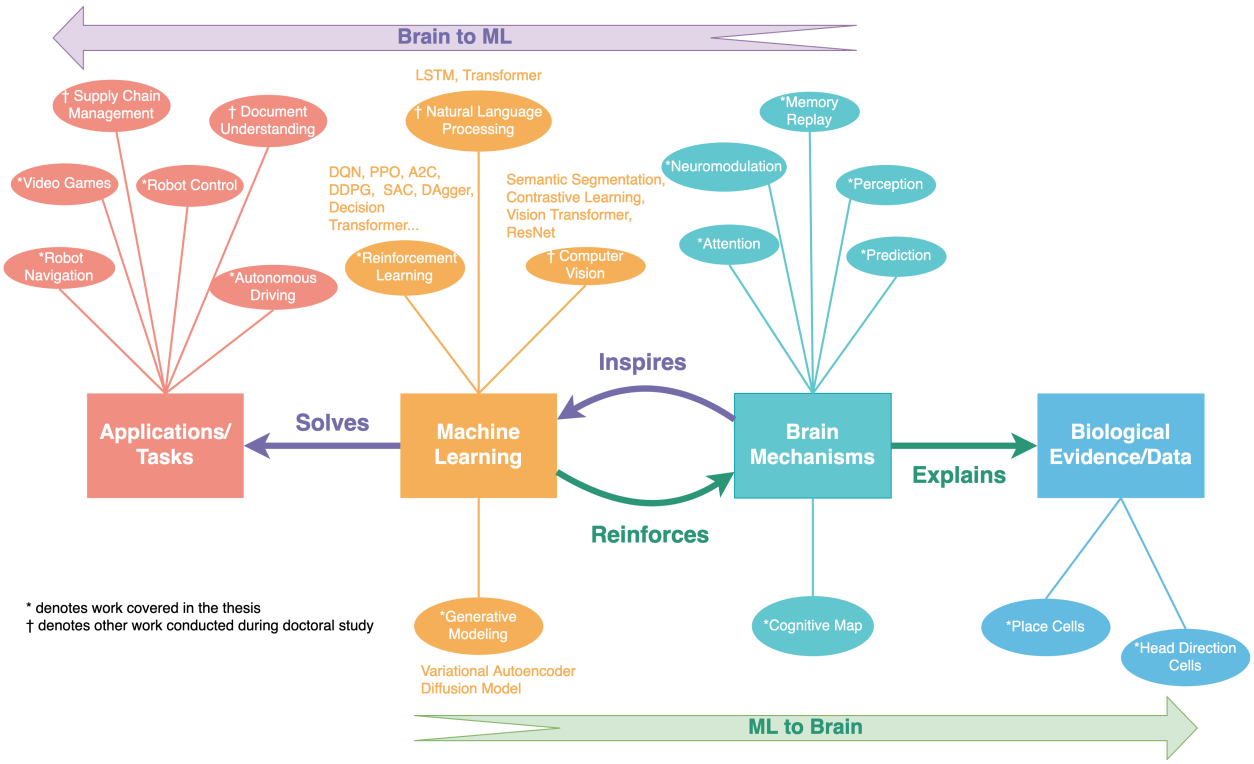


Figure 1.1: Summary of research work conducted during my doctoral study. Four rectangle blocks (Applications/Tasks, Machine Learning, Brain Mechanisms and Biological Evidence/Data) form the base of the diagram. The green and purple arrows represent their relationships (Brain mechanisms inspire machine learning which solves tasks while machine learning also reinforces brain mechanisms which explain biological data). Each of them further derives a series of examples/instances involved in my doctoral study.

from the advancement of machine learning. Machine learning algorithms can be used to analyze complex neuroscience datasets, enabling the identification of patterns and relationships that may not be easily detectable through traditional methods. Moreover, machine learning can reinforce existing theories of brain mechanisms by identifying similar mechanisms in both artificial neural networks and the human brain. By leveraging machine learning, researchers can gain new insights into the cognitive processes and functions, leading to a better understanding of brain.

This dissertation covers a variety of machine learning methods augmented with brain-inspired components and algorithms to address dynamic environments, and one study to understand the cognitive function of spatial perspective transformation with machine learning models.

Much of the work presented in this thesis relies on reinforcement learning. Chapter 2 provides background on AI, machine learning, neuro-inspiration, and reinforcement learning. Chapter 3 shows how an autonomous system augmented with patience-based neuromodulation mechanism could navigate in outdoor environments and adjust behavior based on the context and uncertainty of a situation[Xing et al., 2020]. Chapter 4 demonstrates how prediction and neuromodulation modules allow rapid detection of environment changes and achieve quick adaptation them [Xing et al., 2022c]. Chapter 5 introduces a reinforcement learning method to achieve seamless domain adaptation between multiple visually-different environments, focusing on the perception [Xing et al., 2021]. Chapter 6 introduces a method to improve the explanability and robustness of agent decision making, partially inspired by memory replay[Xing et al., 2023]. Chapter 7 transitions to understanding the cognitive function of spatial perspective transformation using the machine learning method of variational autoencoders (VAEs) [Xing et al., 2022a]. Finally, Chapter 8 discusses future directions and conclusions in the neurorobotics of spatial navigation.

Chapter 2

Background

2.1 Reinforcement Learning

Reinforcement learning is concerned with how agents should take actions in an environment in order to maximize their cumulative rewards. The environment is typically stated in the form of a Markov decision process (MDP), which is expressed in terms of state s, action a and numeric rewards r. At each time step t in the MDP, the agent takes an action a_t in the environment based on current state s_t and receives a reward r_{t+1} following $P(r_{t+1}|s_t, a_t)$ and next state s_{t+1} following $P(s_{t+1}|s_t, a_t)$ (See Figure 2.1). The goal of the agent is to find a policy $\pi(s)$ to choose actions to maximize the discounted cumulative future rewards $r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...$ where γ is the discount factor ranges from 0 to 1.

Because of the MDP setting, the state-action value $Q(s, a)$ could be recursively expressed via the Bellman Equation [Bellman, 1966] which underlies most reinforcement learning algorithms.

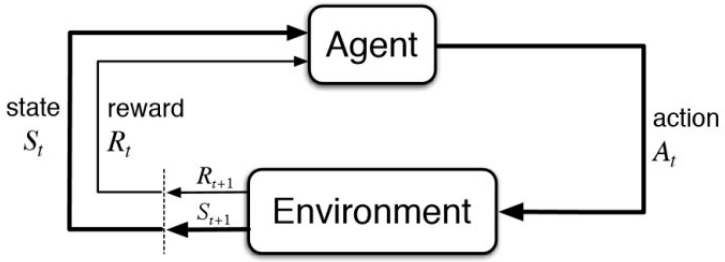


Figure 2.1: Reinforcement learning illustration

$$Q_\pi(s_t, a_t) = r_t + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) Q_\pi(s_{t+1}, \pi(s_{t+1})) \tag{2.1}$$

Based on the Bellman equation, there are two main classes of reinforcement learning algorithms, which are known as model-based and model-free. Model-free reinforcement learning focuses on approximating the value estimation of states or state-action pairs directly from experiences. One option to learn value estimation is Monte-Carlo methods. In Monte-Carlo methods, a batch of complete episodes are first generated and then the empirical mean is calculated from this value estimation. Monte-Carlo methods rely on tracking complete episodes which could be difficult especially when the episodes are very long. Temporal difference learning is proposed to solve this challenge. It could update the value estimation from incomplete episodes with the equation below:

$$V(s_t) = V(s_t) + \alpha(r_{t+1} + V(s_{t+1}) - V(s_t)) \tag{2.2}$$

where value is updated towards the estimated return and α is the learning rate.

Different from model-free reinforcement learning that learns value approximation only, model-

based reinforcement learning has access to or learns a model of the environment. The model contains information of the transition dynamics of the environment which is $P(s_{t+1}, r_t|s_t, a_t)$. The model allows the agent to do planning by thinking ahead, seeing what would happen after different choices and deciding what action to take. When the model is accurate, it helps the agent to do better decision making by allowing it to do planning and also improves sample efficiency since the model could be used to sample experiences for training just like the real environment. However, modeling the environment can be very challenging, especially when the state space is large. As a result, model-based reinforcement learning requires more processing and time to reach a decision. Compared to model-based reinforcement learning, model-free reinforcement learning is faster and takes less computation, but cannot reason over complex problems.

2.2 Neuromodulation

Neuromodulation is a field of neuroscience that focuses on the study of the neural mechanisms that regulate the activity of neuronal circuits in the brain. Neuromodulatory systems, including the noradrenergic (NE), serotonergic (5-HT), dopaminergic (DA), and cholinergic (ACh) systems, track environmental signals and regulate various cognitive, emotional, and physiological processes such as attention, learning and memory, arousal, and mood.

NE neurons exhibit a robust response to both unexpected environmental changes and task-relevant stimuli. This suggests that they play a crucial role in triggering a "network reset", which leads to a broad reorganization of neuronal activity throughout the brain [Bouret and Sara, 2005]. This process enables changes in behavior and cognition, facilitating the adaptation to new circumstances. Xing et al. [2022c] used NE system to track the unexpected uncertainty about the environment to allow rapid adaptation to environment changes.

5-HT neurons is believed to be important not only for regulating anxious behavior and harm aversion, but also having an influence on patience control [Miyazaki et al., 2018]. 5-HT based neuromodulatory system is applied in [Xing et al., 2020] for patience control in mobile robot's autonomous driving and navigation system. In a set of outdoor experiments, we showed how changing the level of patience can affect the amount of time the robot will spend searching for a desired location.

Dopamine neuromodulation is a vital mechanism in the brain that regulates reward, motivation, and motor control. Theoretical and experimental evidence suggests that dopamine neurons encode reward prediction error which is important for reward-based learning such as reinforcement learning. Alternative hypotheses also suggest dopamine neurons respond to salient or novel environmental events to discover novel actions, or contains the uncertainty of alternative actions beliefs.

ACh plays an important role in memory consolidation [Hasselmo and McGaughy, 2004], attention on sensory information [Zou et al., 2020] and uncertainty-mediated inference computations [Yu and Dayan, 2005]. In our project, we used an ACh system to track the expected uncertainty about the environment. Our ACh system was designed such that each environment setting corresponded to a specific ACh neuron. The level of activation of the ACh neuron was directly proportional to the likelihood of the corresponding environment setting being the current state. As a result, the reinforcement learning agent could choose appropriate actions based on the status of the ACh system which is particularly helpful when the environment could dynamically change.

2.3 Attention

Attention is a crucial cognitive process in the human brain that enables us to selectively focus on specific stimuli or tasks while ignoring irrelevant or distracting information which is important due to the constraints of limited processing resources in brain. Attention is frequently studied as arousal or alertness [Oken et al., 2006, Posner, 2008], a form of resource selectively deployed to specific sensory inputs [Zhou and Desimone, 2011, Bichot et al., 2015] and important components in executive control [Miller and Buschman, 2014] and memory [Aly and Turk-Browne, 2017].

Attention mechanism is also widely studied in the field of machine learning for explainability and computation. Saliency maps or attention maps are visualization techniques used to identify and highlight the most relevant or important regions in an input data, such as an image or text. These maps help to provide insights into the inner workings of machine learning models, especially neural networks, by demonstrating which parts of the input data contribute the most to the model's decision-making process. There are several methods for generating saliency maps in machine learning. Gradient-based methods [Simonyan et al., 2013, Springenberg et al., 2014, Selvaraju et al., 2017] compute the gradient of the model's output with respect to the input data and use the resulting gradient values to indicate the sensitivity of the model's prediction to small changes in the input, allowing for the identification of important regions. Perturbation-based methods [Greydanus et al., 2018, Puri et al., 2020, Xing et al., 2022b] systematically perturb or occlude parts of the input data and measure the impact of these modifications on the model's output. Compared to perturbation-based methods, gradient-based methods normally are more computationally efficient but the resulted saliency maps are more blurry. Xing et al. [2023] proposed a technique of distillation with selective input gradient regularization to achieve both computation efficiency and explainability.

Attention mechanisms are also used in deep learning as a computation form to improve the performance of deep learning models. It provides a way for models to weigh and prioritize certain parts of the input data during the processing, enabling them to focus on the most relevant features for a given task. It was first introduced to address the limitations of sequence-to-sequence (seq2seq) models in natural language processing (NLP) tasks, such as neural machine translation [Vaswani et al., 2017]. Traditional seq2seq models rely on fixed-length context vectors to represent entire input sequences, which can lead to information loss and difficulty in handling long sequences. Attention mechanisms were designed to alleviate this issue by allowing the model to dynamically select and focus on specific parts of the input sequence during the decoding process. More recently, the mechanism of self-attention was developed to allow a model to focus on different parts of the input sequence relative to each element within the sequence itself and has been the foundation for many state-of-the-art models in natural language processing, such as BERT [Devlin et al., 2018] and GPT [Brown et al., 2020].

2.4 Generative Modeling

Generative modeling is a powerful approach in machine learning that focuses on learning the underlying structure and patterns of a given dataset. By understanding and representing the data's distribution, generative models can generate new, previously unseen samples that closely resemble the original data. Deep generative models, which leverage the expressive power of deep neural networks, have played a significant role in advancing the state of the art in generative modeling. By approximating complex probability distributions using vast amounts of training data, deep generative models have demonstrated remarkable performance in various applications such as image generation, text generation, text-to-image conversion, speech synthesis and so on.

The successes of deep generative modeling come from the rapid development of modeling approaches such as Variational Autoencoders (VAE), Generative Adversarial Networks (GANs), Normalizing Flows, Diffusion Models and Transformer-based Generative Models. This section focuses on the introduction of VAE as this is the main approach used in my dissertation projects.

VAEs combine concepts from deep learning and probabilistic graphical models to create a flexible, unsupervised learning framework. They can be used to learn a low dimensional representation z of high dimensional data x such as images. The relationship between the data input and the latent encoding vector can be parameterized with θ where $p_\theta(z)$, $p_\theta(x|z)$ and $p_\theta(z|x)$ represent the priority probability of z, likelihood of x given z and posterior probability of z given x. VAEs use $q_\phi(z|x)$ to approximate $p_\theta(z|x)$ and optimize the Evidence Lower BOund (ELBO), which is a lower bound on the log-likelihood of the data. The final objective is:

$$
\begin{aligned}
L_{\mathrm{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\
\theta^*, \phi^* &= \arg\min_{\theta, \phi} L_{\mathrm{VAE}}
\end{aligned}
$$

$p_\phi(z|x)$ and $q_\theta(x|z)$ are represented by two neural networks respectively. The encoder $p_\phi(z|x)$ maps the input data to a lower-dimensional latent space and the decoder $q_\theta(x|z)$ reconstructs the original data from the latent representation. By optimizing the ELBO, VAEs learn to generate diverse and realistic samples while simultaneously learning meaningful latent representations of the data. These latent representations can be used for various downstream tasks, such as data visualization, clustering, transfer learning for supervised tasks and so on.

Chapter 3

Neuromodulated Patience for Robot and Self-Driving Vehicle Navigation

3.1 Introduction

Real-world environments can change due to the season, time of day, construction, or the behavior of other agents. Furthermore, goals, motivations, or context can change due to altered conditions. Uncertainty can arise due to sensor noise, unforeseen obstacles or uncertain goals. An autonomous system needs to cope with these challenges and have the ability to rapidly adapt its behavior based on the current situation.

For successful behavior in a dynamic world, an agent may need to tradeoff between patience and assertiveness. For example, a self-driving car may get stuck at a four-way stop sign

because human drivers are not waiting their turn. A self-driving car that became impatient would eventually assert itself, and move into the intersection. On the other hand, in a dangerous driving situation (e.g., icy roads), an autonomous vehicle may need to slow down and possibly delay its arrival time for safe travel. In this case, patience is a virtue. Or if a search and rescue robot's task is to locate as many injured people as possible, even if it means the robot could run out of energy, patient search would be a priority. In these cases, a signal dynamically regulating the patience, or impatience, of the autonomous system would be beneficial.

Biological inspiration for regulating patience in autonomous systems could be obtained from the mammalian nervous system, which has a number of neuromodulators that regulate context, signal changes, and direct actions. The neuromodulator serotonin (5-HT) is thought to have a role in harm aversion, anxious states, and temporal discounting [Avery and Krichmar, 2017]. Recently, Miyazaki and colleagues showed that optogenetically increasing 5-HT levels caused mice to be more patient, especially when the timing of a reward was uncertain [Miyazaki et al., 2018]. Based on these results, they developed a Bayesian decision model for the probability to wait or quit.

Although great progress has been made in the robotics community for path planning, there are still a number of open issues when it comes to flexible navigation under dynamic conditions [Lavalle, 2011]. Classic path planning algorithms include Dijkstra's algorithm, A Star (A*), and D*. Dijkstra's algorithm uses a cost function from the starting point to the desired goal. A* additionally considers the distance from the start to the goal "as the crow flies" [Hart et al., 1968]. D* extends the A* algorithm by working backward from the goal toward the start position, and can readjust costs, allowing it to replan paths in the face of obstacles [Stentz, 1994]. However, these cost functions are typically fixed or deterministic. Neurobiolgically inspired algorithms have demonstrated the ability to readjust paths depending on cost, such as our work on adaptive path planning [Hwu et al., 2017], and Erdem

and Hasselmo's work that demonstrated the ability to take shortcuts [Erdem and Hasselmo, 2012]. The above algorithms do not consider motivation or context, and do not reflect the flexibility observed in animal navigation.

In order to add context and flexibility to path planning, we apply the Miyazaki et al. [2018] rodent model of patience to a ground robot. Specifically, our robot navigates through a series of waypoints. The level of 5-HT dictates how patiently the robot will search for a waypoint. We show that changing the 5-HT level can have dramatic effects on the robot's behavior. Such a system may be beneficial for adjusting autonomous behavior depending on the context and uncertainty of a situation.

3.2 Methods

3.2.1 Navigation Task

Robot navigation tasks were carried out in two different outdoor parks with varying terrain and features. Figure 3.1 shows satellite images of the two parks. Waypoints were GPS coordinates placed on sidewalks in the park. The park on the left of Figure 3.1, Encinitas Community park, was relatively flat. Waypoints were placed along the perimeter of the test area on either sidewalks or the paved parking lot. In the middle of the test area was a grassy region with some trees. The park on the right of Figure 3.1, Aldrich park at the University of California, Irvine, was hilly with numerous obstacles (e.g., bushes, benches, and buildings). It should be noted that the Aldrich park test area was in a sunken bowl surrounded by tall buildings and trees. These features made GPS signals unreliable. For this reason, a road following algorithm, which will be discussed below, was introduced to assist with navigation. The waypoints were placed on the sidewalk that surrounded the inner grassy region.

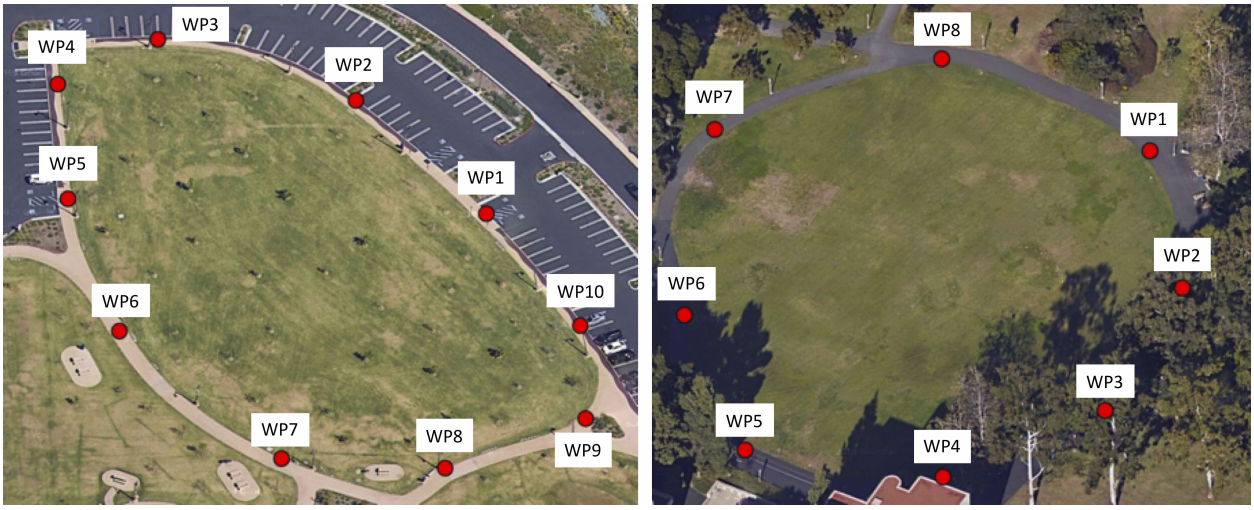


Figure 3.1: Parks where robot navigation experiments were carried out. The left is an image of Encinitas Community park and the right is an image of Aldrich park at the University of California, Irvine. The labels denote the waypoints (i.e., WP1...WP10). Waypoints were approximately 50-60 meters apart. Imagery from Google Maps, 2019.

In both parks, the robot's task was to proceed to each waypoint in order. If the robot became impatient, it would skip searching for the present waypoint and randomly choose a future waypoint. However, the robot had to reach the last waypoint for a trial to be complete.

3.2.2 Robot and Software Design

For the robot experiments, we used the Android-Based Robotic platform [Hwu et al., 2017], a mobile ground robot constructed from off-the-shelf commodity parts and controlled through an Android smartphone (see Figure 3.2). An IOIO-OTG microcontroller communicated with an Android smartphone via a Bluetooth connection and relayed motor commands to a separate motor controller for steering the Dagu Wild Thumper 6-Wheel Drive All-Terrain chassis. Three ultrasonic sensors, which were used for obstacle avoidance, were connected to the robot through the IOIO-OTG. A software application, which controlled the robot, was written in Java using Android Studio and deployed on a Google Pixel XL smartphone. The application utilized the phone's built-in camera, accelerometer, gyroscope, compass, and

(a) Android Based Robot in Encinitas Park.

(b) Android Based Robot in Aldrich Park.

Figure 3.2: Android Based Robot used for the experiments.

GPS for navigation.

For waypoint navigation, a GPS location was queried using the Google Play services location API. The bearing direction from the current GPS location of the robot to a desired waypoint was calculated using the Android API function bearingTo. A second value, the heading, was calculated by subtracting declination of the robot's location to the smartphone compass value, which was relative to magnetic north. This resulted in an azimuth direction relative to true North. The robot traveled forward and steered in attempt to minimize the difference between the bearing and heading. The steering direction was determined by deciding whether turning left or turning right would require the least amount of steering to match the bearing and heading. The navigation procedure continued until the distance between the robot's location and the current waypoint was less than 20 meters, at which point the next waypoint in the list was selected.

3.2.3 Waypoint Navigation and Model of Neuromodulated Patience

The robot proceeded through a list of waypoints as described above. However, if the robot became impatient, it skipped the present waypoint and randomly chose a waypoint closer to the final destination.

The likelihood to skip a waypoint was based on the Bayesian Decision Model given by [Miyazaki et al., 2018]. Specifically, we calculated the probability to wait given the time elapsed:

$$p(wait|t) = \frac{1}{1 + exp^{\beta \cdot 5HT \cdot L(t)}}, \tag{3.1}$$

where β was equal to 50, and L(t) was the likelihood of reaching the waypoint at time t, and 5HT denoted the serotonin level. The likelihood was calculated with a Normal cumulative distribution function having a mean of 40 seconds and a standard deviation of 20 seconds. The likelihood function was multiplied by a scalar that represented the probability of receiving a reward. As in Miyazaki et al. [2018], we assumed that increasing 5-HT levels caused an overestimation of the prior probability. Therefore, in our experiments low 5-HT equated to a probability of a reward of 0.50 and high 5-HT equated to probability of a reward of 0.95 (see Miyazaki et al. [2018] for details). Figure 3.3 shows the resulting probability to wait, $p(Wait|t)$, curves.

The $p(Wait|t)$ curves in Figure 3.3 were used to decide whether to keep searching for a waypoint or to forego the desired waypoint and choose another. A random number between 0 and 1 was generated and if the number was greater than $p(Wait|t)$, where t was the time

elapsed that the robot had been searching for a waypoint, the robot stopped searching for this waypoint. A new waypoint was randomly chosen that was closer to the final destination. Note that if the robot was searching for the final destination waypoint or for a waypoint after a skip, the $p(Wait|t)$ curve was not referenced. That is, the robot had to reach the shortcut waypoint or had to reach the final waypoint for a successful trial. See Algorithm 1 for implementation details.

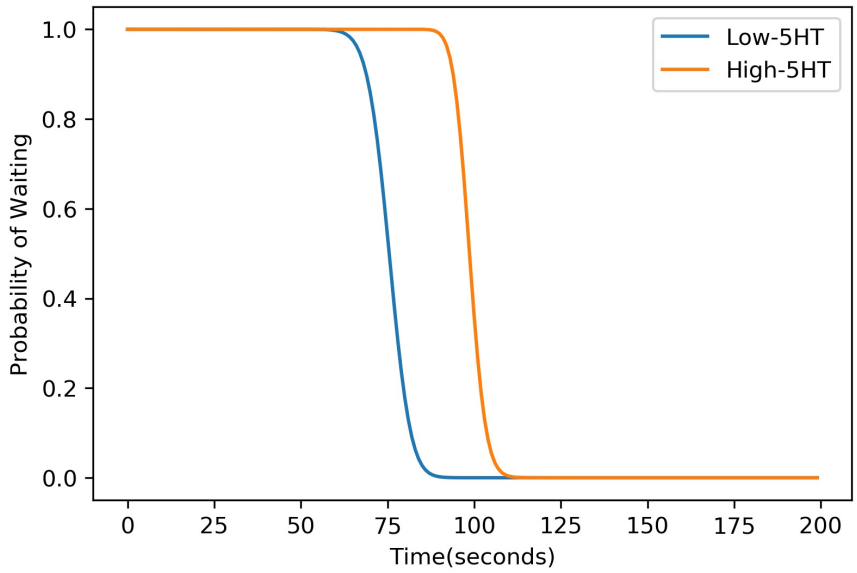


Figure 3.3: Probability of waiting function. Higher 5-HT levels shifted the curve to the right resulting in longer wait times.

3.2.4 Road Following with Deep Reinforcement Learning

A road following algorithm based on deep reinforcement learning was used in the experiments carried out in Aldrich park. This became necessary due to poor GPS reception in this environment. We used a Deep Q-Network (DQN) for online learning of a driving policy on the Aldrich park sidewalks [Mnih et al., 2013].

Algorithm 1 Waypoint Navigation with Neuromodulated Patience

Input: GPS and compass readings, N waypoints Initialize waypoint index $w = 0$ Initialize time count $t = 0$ Initialize $shortcut$ = false Initialize $finished$ = false **while** *not* $finished$ **do**

```
    get the current GPS and compass readings if robot is within 20m of waypoint(w) then
        if w == N then
            finished = true break
        else
            w = w + 1  t = 0
        end
        shortcut = false
    end
    if not shortcut and w != N then
        generate a random number rand_num and update p(Wait|t) if rand_num > p(Wait|t) then
            update w with a random integer in the range of [w + 1, N)  t = 0  shortcut = true
        end
    end
    if not shortcut and in Aldrich park and on road then
        move forward toward waypoint(w) based on road following algorithm
    else
        use GPS and compass to get bearing to waypoint(w)  navigate toward waypoint(w)
    end
    t = t + 1
end
```

Road Following DQN States and Actions

In reinforcement learning, an agent is acting in an environment. At each time step t, the agent chooses an action $a_t \in A$ in response to the current state $s_t \in S$. The system makes the transition from s_t to s_{t+1} with a reward r_t based on the reward function $R(s_t, a_t)$. The goal of reinforcement learning algorithms is to learn a policy that maps a state s to an action a, such that the expected sum of rewards $\mathbb{E}_\pi[\sum_t^\infty \gamma^t r^t | s_t, a_t]$ is maximized where π is the agent's behavior function. $\gamma \in [0, 1]$ is a discounting factor used to penalize the rewards in the future. As a value-based deep reinforcement learning method, the DQN learns a state-action value function $Q_\theta(s, a)$ which outputs the expected discounted sum of future rewards that will be received by following the policy. Some recent works used deep reinforcement learning in robot navigation tasks Kahn et al. [2018], Faust et al. [2018], but all of them are set in ideal indoor environments. To the best of our knowledge, our project is the first work that trained the robots to navigate in complicated outdoor environments with deep reinforcement learning.

In our experiments that utilized road following, the agent was the Android-Based Robot and the environment was Aldrich Park (see figure 3.6). The state was represented by an annotated camera image, as will be described in Section 3.2.4, from the smartphone that was mounted on the robot. The reward was either 0.5 when the robot stayed on road or 0 when the robot went off road. In the beginning of each training episode, the robot was initialized in the center of the road. When the robot was not on road, the episode ended and the robot was reset to the center of the road for the next episode. In each step, the robot moved forward for 0.6 seconds with a constant speed but a different steering angle ranged from sharp left to slight left to straight to slight right and to sharp right. During the training, the robot was reinforced by staying on the road. After around 15 episodes and 2 hours of training, roughly 2000 training steps, the robot learned to follow the road.

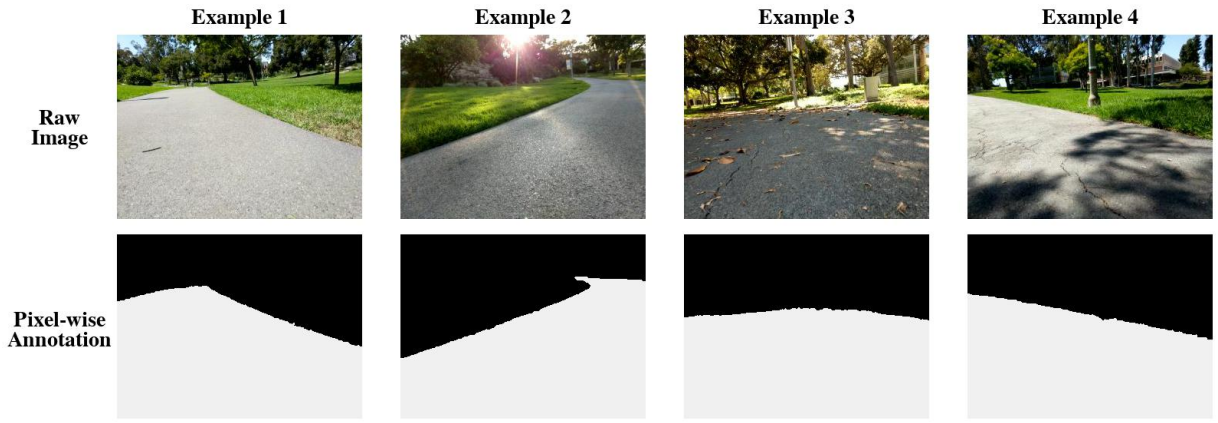


Figure 3.4: Examples of raw images and pixel-wise annotations in Aldrich Park dataset. The size of each image is 320x240 pixels and each pixel has a label of either 1 (road) or 0 (non-road). In visualized annotations (lower half), the non-road portion is shown in black and the road portion is in gray.

Semantic Segmentation of Images

To evaluate the states of the robot in the environment and then generate rewards for the deep reinforcement learning module, we used ENet [Paszke et al., 2016], a pixel-wise real-time semantic segmentation neural network. ENet labeled each pixel of the image as road or non-road. We used middle-bottom portion of the segmented image to evaluate if the robot was on road. The image size in the experiment is 320x240 pixels and the size of middle-bottom portion for evaluation is 80x32 pixels. If most pixels in that portion were labeled as road, we judged that the robot was on road. Otherwise, the robot was thought to be off-road.

The environment of Aldrich Park and camera setting in this project were very different from those of popular datasets such as Kitti [Menze and Geiger, 2015] where road detection was also involved. Therefore, we created a scene understanding dataset for the robot from data collected in Aldrich Park. Smartphone camera frames were collected in Aldrich Park at different times (i.e., 2pm to 7pm) of day. We selected 418 distinct and representative pictures and performed binary (road and non-road) pixel-wise labeling for these using the PixelAn-

notationTool from [Bréhéret, 2017]. Figure 3.4 shows examples of semantic segmentation taken from the Aldrich Park dataset. The ENet model trained on the Aldrich Park dataset allowed us to rapidly label road and non-road portions of a scene and generate rewards for the deep reinforcement learning module.

Besides its necessity for reward generation, the semantic segmentation module provided two other benefits. First, the segmented observation, which was fed to the deep reinforcement module, removed noisy information from the original image and kept the most important features (road or non-road). This simplified the task for deep reinforcement learning and thus the training of the DQN was faster. Second, the semantic segmentation module increased the generalization and adaptability of the self-driving navigation to handle dynamic characteristics of outdoor environments such as lighting changes due to time of day or weather. Examples in Figure 3.4 show some of the various lighting conditions in the park. Without the semantic segmentation module, the DQN trained at 2pm could not work at 7pm because sunlight differences. To solve this problem without the semantic segmentation module, we would have needed to train under all different environment situations, which would be time consuming and would need to deal with potential problems such as catastrophic forgetting. Another case that demonstrates the advantage of semantic segmentation is that, the robot could avoid a pedestrian automatically because the pedestrian would be labeled as non-road and the robot would try to stay on road. The robot trained without semantic segmentation could not achieve this and would instead take random action since the appearance of a pedestrian was a novel state for it. By separating the scene understanding task from reinforcement learning, semantic segmentation enables faster training and better generalization capability [Hong et al., 2018].

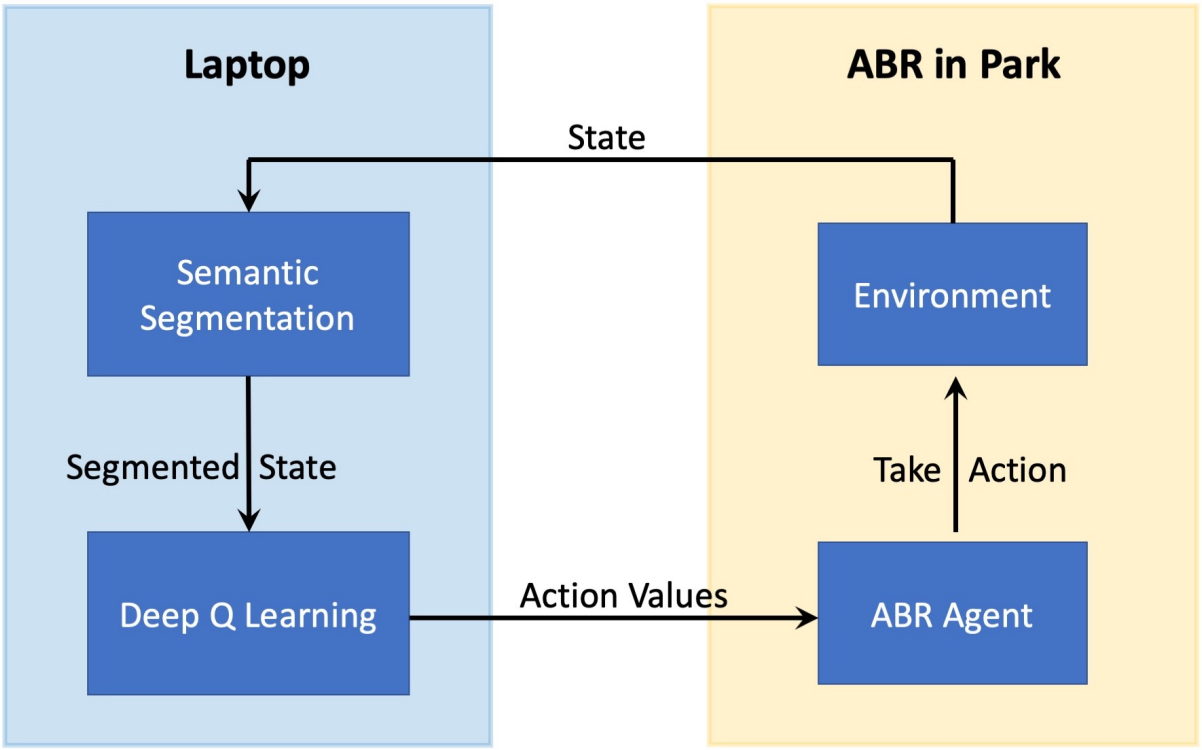


Figure 3.5: High level illustration of the data pipeline for the road following algorithm. Images from the Android Based Robot's (ABR) smartphone camera are sent to a nearby laptop via a socket. The laptop runs a deep reinforcement learning algorithm, which rewards staying on the road, and generates steering actions for the robot.

Road Following Data Pipeline

Figure 3.5 shows the data pipeline. The Android Based Robot took pictures with the smartphone's camera. Using a WiFi "hotspot", the image was sent to a nearby laptop, which performed real-time image segmentation of "road" versus "non-road". The laptop also ran a deep reinforcement learning network, based on the DQN, which processed the image and outputted action values used by the agent to choose actions. The actions ranged from sharp left to slight left to straight to slight right and to sharp right. The reward was also based on the segmented state. A detailed illustration of the road following deep reinforcement neural network is given in Figure 3.6. The laptop took about 400 ms to process the information, generate an action, and update the network. This was adequate for online learning in real-time.

The robot learned to follow the road after roughly 2000 training steps. The road following algorithm was used in Aldrich park where waypoints were set along the sidewalk that surrounded a hilly grass region (Figure 3.1, right). The robot could move to the next waypoint by following the road. In the present experiments, we segmented road and non-road. But, potentially, we could also segment people, trees, benches, etc. These object classes could be used as further inputs for training the network and implementing more complex behavior.

3.3 Results

Two sets of robot navigation experiments were carried out. One set was in the Encinitas Community park (see Figure 3.1 left) and the other was in Aldrich park (see Figure 3.1 right). In both cases, the robot navigated through a set of waypoints with low and high 5-HT levels. In Aldrich park, the navigation experiments were carried out with road following activated.

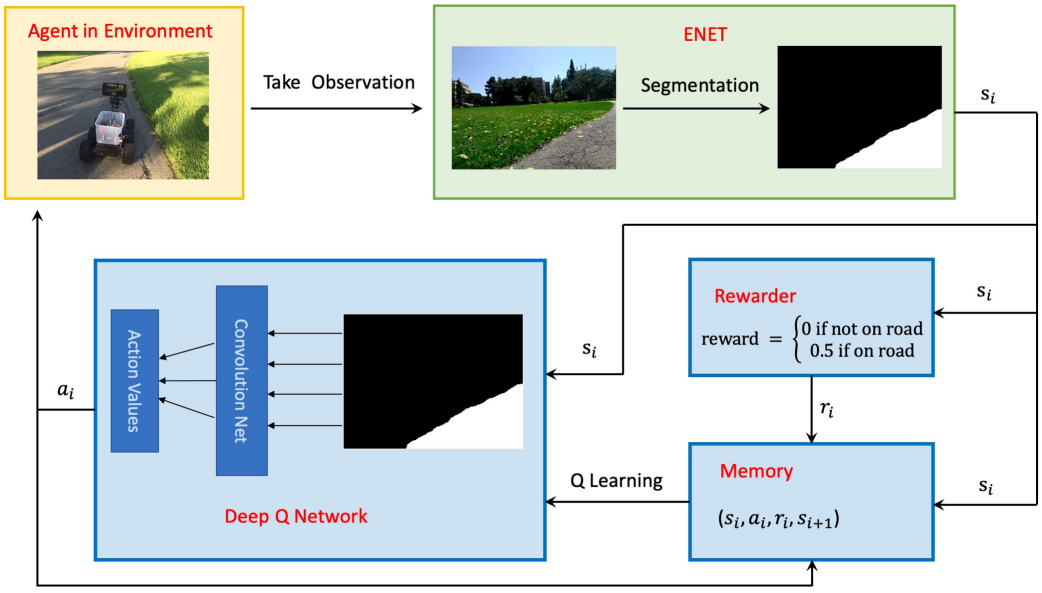


Figure 3.6: Detailed illustration of the data pipeline for the road following algorithm. ENet was used to segment road from non-road Paszke et al. [2016]. The network gave a positive reward for actions that kept the robot on the road and a penalty for actions that caused the robot to go off road. Training was based on a DQN reinforcement learning paradigm Mnih et al. [2013]. Training and testing were carried out online in Aldrich park.

3.3.1 Waypoint Navigation in Encinitas Community park

We ran 6 trials for low 5-HT and 6 trials for high 5-HT in the Encinitas Community park (see Figure 3.7). The waypoints were roughly 50-60 meters apart. In Figure 3.7, each marker denotes the GPS location from the smartphone when the robot was within 20 meters of a waypoint (different colors denote different trials). Note that this reading could vary dramatically due to GPS inaccuracies.

The level of 5-HT affected the robot's patience in finding a waypoint. Over the 6 trials, 9 waypoints were skipped when 5-HT was low, but only 2 waypoints were skipped when 5-HT was high. The average time before skipping a waypoint was 68 seconds for low 5-HT and 97 seconds for high 5-HT (see Table 3.1). These experiments demonstrated how this model could change route planning behaviors.

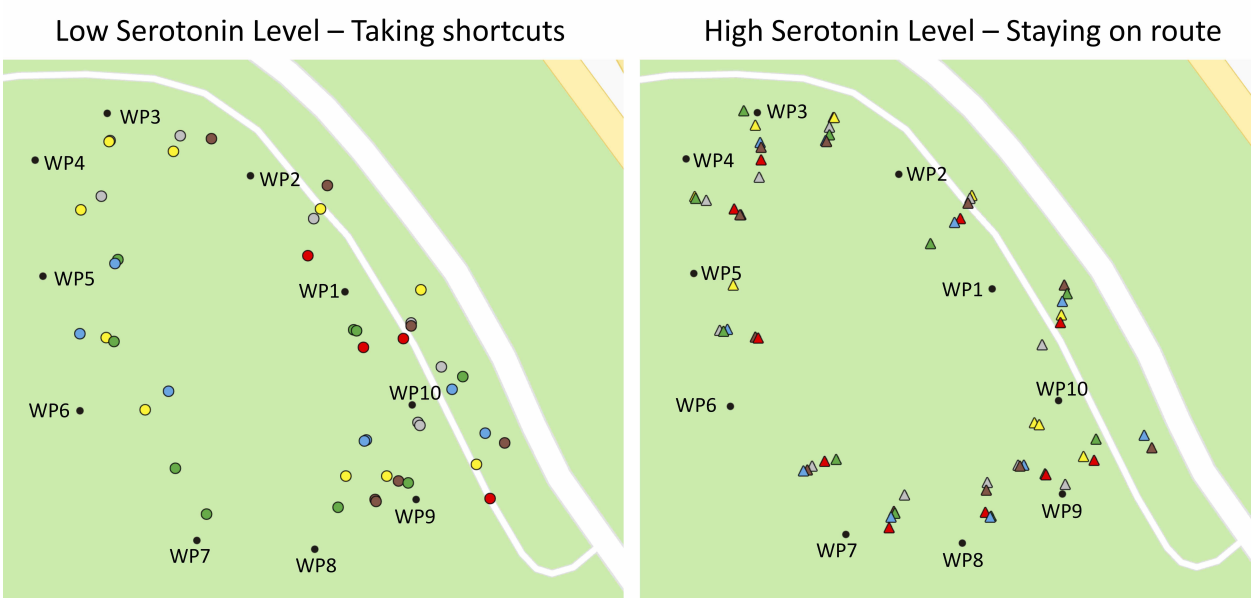


Figure 3.7: Robot navigation trials in the Encinitas Community park. The black dots are the waypoint destinations (WP1 – WP10). There were 6 trials. Each color represents an individual trial. Each colored marker denotes the robot reaching a waypoint.

Figure 3.8 shows all the GPS readings from two representative trials, one with high 5-HT and the other with low 5-HT. In the high 5-HT trial, the robot reached every waypoint. In the low 5-HT trial, the probability to wait was exceeded for reaching Waypoint 6 after 69 seconds and the robot skipped to Waypoint 9. A video of the robot performing waypoint navigation with low 5-HT can be found at: `https://youtu.be/6EcNchTGLKw`, and a video of the robot performing waypoint navigation with high 5-HT can be found at: `https://youtu.be/q_m0gbVN6UE`.

Table 3.1: Results of High/Low 5-HT Modulating Navigation

	Encinitas Park (6 trials)		**Aldrich Park (5 trials)**	
	High 5-HT	Low 5-HT	High 5-HT	Low 5-HT
Navigation Time (s)	525.521	413.549	414.905	389.625
Shortcuts	0.3	1.5	0.0	1.4
Waypoints Reached	9.67	6.5	8.0	6.0

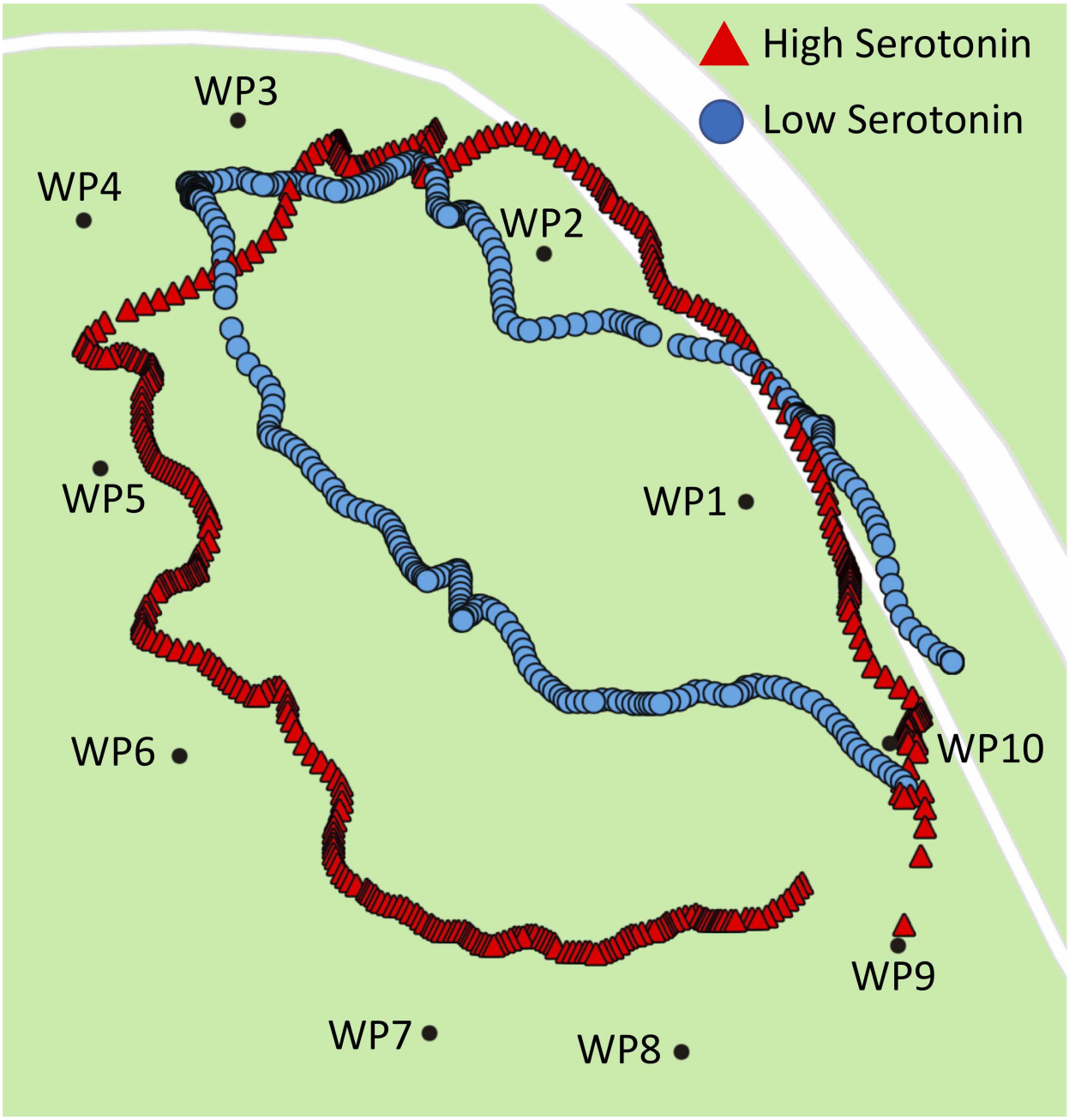


Figure 3.8: Two representative navigation trials in the Encinitas Community park. All the GPS points are shown. The markers in the figure correspond to the grey markers in Figure 3.7.

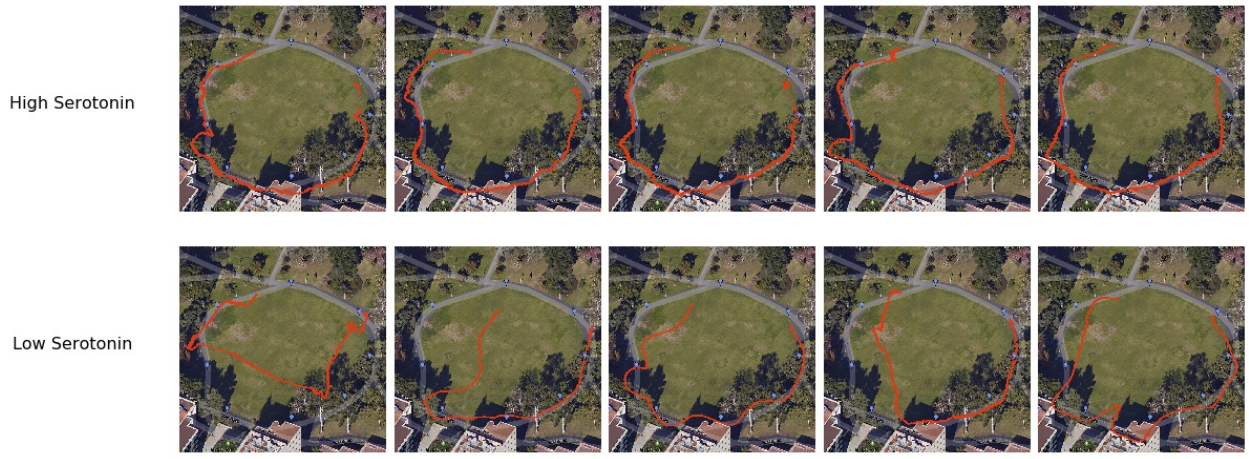


Figure 3.9: Five trials in Aldrich park with high 5-HT (upper figure) and five trials in Aldrich park with low 5-HT (lower figure). Red traces are drawn from GPS readings from the phone mounted on the robot. The robot went across all waypoints one by one in trials with high 5-HT and took multiple shortcuts when 5-HT was low.

3.3.2 Waypoint Navigation in Aldrich park

We ran 5 high 5-HT trials and 5 low 5-HT trials in Aldrich park (see Figure 3.9). Since the area is sunken in a bowl surrounded by tall buildings and trees, the GPS readings were highly inaccurate. In particular, the robot had difficulty finding Waypoints 2 and 3 due to the poor GPS signal. As a result, we introduced the road following algorithm described in Section 3.2.4, which helped the robot stay on the sidewalk and increased the likelihood of finding a waypoint within the probability of wait constraint. During road following based navigation, the robot moved towards the waypoint by following the road. Since the waypoints were placed along the outer ring of the test area, the robot tended to move closer to the next waypoint by following the road. When the robot decided to take a shortcut because of being impatient, the movement of the robot was based on GPS because the shortcuts took the robot off the road and over the grassy interior of the test area. Once the shortcut waypoint was reached, the road following algorithm took over again.

It should be noted that during road following, the robot took longer to complete the course with high 5-HT (i.e., 420 seconds on average) than with low 5-HT (i.e., 390 seconds on

average) in which shortcuts were taken. However, since the robot was traveling over smoother terrain with high 5-HT, it reached more waypoints and took less energy than when it took shortcuts with low 5-HT (see Table 3.1). A video of the robot navigating using road following can be found at: `https://youtu.be/DixOxO2UafQ`.

These results show the benefits and the tradeoffs associated with being patient versus being impulsive during navigation. In both two parks, when 5-HT was high, the robot was more patient when navigating towards waypoints, which meant it took less shortcuts and reached more waypoints, but at the cost of taking longer to complete a trial (see Table 3.1).

3.4 Discussion

In the present paper, we showed how a concept from behavioral neuroscience could be applied to robot navigation and possibly self-driving vehicles. It has been shown that 5-HT in the brain affects impulsiveness in an animal's behavior [Miyazaki et al., 2018]. The present model applied this idea to waypoint navigation in autonomous robots. Specifically, we showed that simulating high 5-HT led to increased search time for a desired location and that simulating low 5-HT led to an increase in calling off the search for some waypoints. Even under high 5-HT conditions, if a waypoint was particularly difficult to find or there were environmental challenges, there was a limit to how long the robot would try to reach a desired location (see Figure 3.9). Our results showed that neuromodulated patience led to flexible behaviors, which are not typically found in traditional navigation solutions [Lavalle, 2011, Stentz, 1994].

The goal of the present algorithm and demonstrations was not to achieve some benchmark, but rather to suggest a neurobiologically inspired strategy that could complement other navigation systems. The present approach could be applied to biomimetic navigation systems [Milford and Schulz, 2014, Gaussier et al., 2019], as well as engineering approaches to navi-

gation [Urmson et al., 2003, Wang et al., 2008]. In general, the probability to wait suggests a level of urgency in the overall system. We imagine this could be applied to a number of tasks where resource allocation is time critical.

Furthermore, the probability of waiting could be associated to some internal parameter in the system (e.g., battery level or prioritizing goals). Presumably, the impulsiveness signal in the rodent is closely tied to its natural foraging behavior. The animal will search for food, but the time it will search depends on the food value and on the uncertainty of the food resource. Such considerations could be beneficial for a robot navigation system or for a self-driving vehicle.

The patience-based neuromodulated navigation algorithm adds another dimension to the present navigation system. By giving the robot an alternative to point-to-point navigation, the robot now must weigh the cost of staying on a smooth and reliable road that may take longer to travel versus traversing over rough terrain that may be shorter but takes more energy and could be potentially harmful to the robot. Since the deep reinforcement learning introduced here is designed for online learning, these costs could be learned along with the rewards for staying on the road. Ideally, the deep reinforcement learning algorithm could set the 5-HT level dynamically.

The present algorithm is a step towards a complete navigation or self-driving system that takes inspiration from neurobiology and behavioral neuroscience.

Chapter 4

Adapting to Environment Changes Through Neuromodulation of Reinforcement Learning

4.1 Introduction

Reinforcement learning (RL) is a learning technique that enables agents to select the appropriate actions in an environment to maximize cumulative reward. In recent years, by combining RL with deep neural networks, deep RL has achieved success in a wide range of applications such as games [Silver et al., 2016, Mnih et al., 2015], robot control [Akkaya

et al., 2019, Lillicrap et al., 2015] and mobile autonomous driving [Kahn et al., 2021, Xing et al., 2020]. Despite these successes, RL still faces challenges when applied to more practical scenarios such as dynamic environments that contain uncertainty [Nagata et al., 2022].

In this work, we investigate how an RL agent can cope with uncertainty in dynamic environments. There are two challenges for the agent. First, the agent needs to rapidly detect the environmental change. Without rapid detection, the knowledge learned before the change could be tainted by the different reward feedback received by the agent after the change. Furthermore, not recognizing the change could result in performance drops. Second, the agent needs to remember the knowledge learned under each environment setting. This requires the agent to not only detect the environment change but also identify whether the changed environment has been interacted with before. When facing a familiar environment setting, the agent needs to recall the knowledge and avoid learning from scratch again.

To address the challenges above, we take inspiration from humans and animals. Decision making is a core competency for humans and animals to survive in the environment. In the past several decades, neuroscience research in decision making tasks support the idea that the brain uses a form of reinforcement learning to shape decision making [Montague et al., 1996, Schultz et al., 1997, Hare et al., 2008, Parker et al., 2016]. However, humans and animals face a similar problem of adapting to environment changes as the real world is normally uncertain. We suggest that brain's neuromodulatory system plays an important role in coping with this uncertainty.

In prior work, we investigated how neuromodulated neural networks could rapidly adapt to goal changes in classification tasks [Zou et al., 2020]. It was based on a Bayesian model of neuromodulation to track the uncertainty of interactions with the environment [Yu and Dayan, 2005] in which the cholinergic (ACh) system tracked expected uncertainty (i.e., the known degree of unreliability of predictive relationships in the environment) and the noradrenergic (NE) system tracked unexpected uncertainty (i.e., large changes in the environment

that violate prior expectations). In addition, the NE system caused a rapid adaptation to goal changes by triggering a 'network reset' [Bouret and Sara, 2005, Grella et al., 2019].

The present work extends Zou et al. [2020]'s goal-driven perception model to RL agents, which must recognize changes to their reward function and adapt appropriately. Inspired by the neuromodulation systems described above, we developed a reinforcement learning system in which ACh system tracks the expected uncertainty of the current reward function while NE system tracks the unexpected uncertainty which could increase when prior actions no longer produce rewards. These two systems allow the agent to detect task changes rapidly and identify whether a task is novel or not and thus decide whether and which knowledge should be recalled. We show how this approach can improve the performance of RL agents in a Gridworld environment and a MuJoCo walking robot.

4.2 Problem

In this work, we focus on the problem of how to enable reinforcement learning agents to rapidly adapt to changes in the environment.

We focus on how to adapt when feedback from the environment changes of the reward function R while other elements including state space S, action space A and state transition function T remain unchanged. Here we define a task as a MDP:

$$Task_i =< S, A, T, R_i > \tag{4.1}$$

where i is the identity of the task. All tasks share the same $< S, A, T >$ while each task has its own R_i. The environment change could be demonstrated as task switching in a task

sequence. As noted below, we define the environment to be composed of a sequence of tasks.

$$Env = [Task_1, Task_2, Task_3, ...Task_N] \tag{4.2}$$

where N is the number of tasks in the sequence. Note that $Task_i$ and $Task_j$ may share the same reward function, in which case $R_i = R_j$. This means a given task can occur multiple times in the sequence, and so the agent needs to learn the task and re-utilize the learned knowledge when exposed to the task again.

4.3 Method

We developed a bio-inspired neuromodulatory system to track the uncertainty of the environment which facilitates rapid adaptation to environment changes. Similar to the model in Zou et al. [2020], our system is composed of ACh and NE neuromodulatory systems. We introduce their underlying equations below.

4.3.1 ACh and NE Neuromodulation

In our system, the ACh system has K ACh neurons where K is the number of tasks the agent has detected and initialized as 1. As more tasks are detected by our neuromodulatory system, the value of K increases accordingly. We use a vector to represent the activity of ACh neurons, where ACh_i represents the uncertainty of $Task_i$. The higher ACh_i is, the more certain that the current task is $Task_i$. A task change is detected when the maximum ACh value is below a threshold ch_{change}.

$$Task\_Change = \begin{cases} True & \text{if } \max(ACh) < ch_{change} \\ False & \text{otherwise} \end{cases} \tag{4.3}$$

Once a task change is detected, the agent needs to judge whether the task is a task that has been learned before and thus previous knowledge should be recalled or the task is novel and needs to be learned from scratch. We call this step a 'task match'. A task match should happen when the agent has low uncertainty on one task and high uncertainty on all other tasks, as described below.

$$P_i = \frac{exp(\beta * ACh_i)}{\sum_j^K exp(\beta * ACh_j)} \tag{4.4}$$

$$Task\_Match_i = \begin{cases} True & \text{if } P_i > p_{match} \\ False & \text{otherwise} \end{cases} \tag{4.5}$$

where P_i represents the certainty that the new task is $Task_i$. Once P_i goes above the threshold of p_{match}, the system matches the new task with $Task_i$ and $Task\_Match_i$ is set as True. The softmax function in Equation 4.4 allows the agent to take the complete ACh system into consideration in task match.

In addition to the task change detection and task match based on ACh system, we use the NE system to decide whether a new task is novel. The NE system consists of one NE neuron whose activity represents the unexpected uncertainty on the environment. The activity of

the NE neuron increases when a task change is detected by the ACh system while the new task cannot be matched with previous tasks. When the activity of the NE neuron goes above a threshold $ne_{threshold}$, the new task is believed to be novel.

$$Task\_Novel = \begin{cases} True & \text{if } NE > ne_{threshold} \\ False & \text{otherwise} \end{cases} \tag{4.6}$$

4.3.2 Update of ACh and NE System

Since the only difference between tasks is the reward function R, the uncertainty about the environment and the update of our neuromodulatory system are based on the difference between the reward feedback expected by the agent and the actual reward, which is called reward prediction error in this work. To produce the reward expectation, the agent learns a reward predictor $\tilde{R}_i$ to approximate R_i for each $Task_i$. For a set of $< s, a, r >$, the reward prediction error is defined as the difference between the predicted reward $\tilde{R}_i(s, a)$ and the true reward r is

$$RPE_i = \tilde{R}_i(s, a) - r \tag{4.7}$$

The expected uncertainty (ACh neuron activity) increases when the reward prediction error is high and decreases when the reward prediction error is low. To simulate this, we use RPE_{mean} and RPE_{std} to track the running mean and standard deviation, respectively, of the reward prediction error and update the ACh system as follows:

$$Expected_i = \begin{cases} True & \text{if } abs(RPE_i - RPE_{mean}) < k * RPE_{std} \\ False & \text{otherwise} \end{cases} \tag{4.8}$$

$$ACh_i = \begin{cases} min(ch_{max}, ACh_i * ch_{expected}) & \text{if } Expected_i \\ max(ch_{min}, ACh_i * ch_{unexpected}) & \text{otherwise} \end{cases} \tag{4.9}$$

where $Expected_i$ represents whether the reward for $Task_i$ is expected by the agent. If so, ACh_i will increase. Otherwise, it will decrease. The hyperparameter k in Equation 4.8 controls the strictness of $Expected_i$. $ch_{expected}$ and $ch_{unexpected}$ are scaling factors used to increase or decrease ACh neuron activities while ch_{max} and ch_{min} represents their maximum and minimum values.

The NE system tracks the unexpected uncertainty of the environment and can be used to detect novel tasks that have have not been observed before. It is updated when a task change is detected and increases when the task match is incorrect. If the current task is matched with an old task, then the unexpected uncertainty is resolved and the NE neuron is reset.

$$NE = \begin{cases} min(ne_{max}, NE * ne_{unmatched}) & \text{if not any } Task\_Match \\ ne_{init} & \text{otherwise} \end{cases} \tag{4.10}$$

where $ne_{unmatched}$ is a hyperparameter bigger than 1 that is used to increase the NE neuron activity. ne_{max} and ne_{init} represent the maximum and initial values of the NE neuron.

4.3.3 The Complete System

The complete system of our work includes an agent conducting reinforcement learning and a neuromodulatory system that helps it track the uncertainty of the environment and adapt to environment changes. The agent needs to remember the knowledge learned in each task. When a task is encountered again, the agent needs to conduct a task match based on the neuromodulatory system and reactivate the knowledge once it's matched to avoid learning from scratch again. In this work, the knowledge includes the RL policy and reward predictor that the agent learns for each task. We describe the complete system with the pseudocode shown in Algorithm 2.

4.4 Experiments

We conducted two experiments to demonstrate the efficacy of our neuromodulatory system for RL applications: 1. Gridworld (Fig. 4.1) and 2. bipedal walking with a MuJoCo robot (Fig. 4.2).

The first experiment is based on a Gridworld Environment [Chevalier-Boisvert et al., 2018]. In this grid-based environment, there are four objects with unique colors where the red color represents the agent while the green, blue and yellow colors represent objects that can be picked up. The agent needs to navigate in the grid world and pick up the correct object. Based on the target object to pick up, we define three tasks named as pickup-green, pickup-blue and pickup-yellow. The agent receives non-zero reward only when picking up an object. The reward is 1 if the agent picks up the correct object and is -1 if the wrong object is picked up.

The second experiment is conducted on a MuJoCo simulated bipedal walker robot. DeepMind control suite [Tassa et al., 2018] contains three walker-based tasks including walker-

Algorithm 2 Reinforcement Learning with Neuromodulatory System

Input: $ch_{init}, ch_{max}, ch_{min}, ch_{expected}, ch_{unexpected}, \beta, k, ne_{max}, ne_{init}, ne_{threshold}$ $ne_{unmatched}, max\_step$

Init: $task \leftarrow 1, ACh_1 \leftarrow ch_{init}, NE \leftarrow ne_{init}, task\_change \leftarrow False, step \leftarrow 0, K \leftarrow 1$

while *step ¡ max_step* **do**

- Agent selects action and receives feedback from the environment
- step ← step + 1
- **if** *not Task_Change* **then**
  - Agent stores experiences for RL training
  - Compute the reward predictor error RPE_{task} `// Equation 4.7`
  - Update ACh system `// Equations 4.8,4.9`
  - Update $Task\_Change$ based on ACh system `// Equation 5`
  - **if** *not* $Task\_Change$ **then**
    - Update RPE_{mean} and RPE_{std}
    - Train the RL agent
    - Train the reward predictor
  - **else**
    - Save the learned knowledge for *task*
- **else**
  - **for** $i \leftarrow 1$ **to** K **do**
    - Compute $Task\_Match_i$ `// Equation 6, 7`
    - **if** $Task\_Match_i = True$ **then**
      - $task \leftarrow$ i
      - $Task\_Change \leftarrow$ False
      - reactivate the saved knowledge for task i
      - reset neuromodulation system
  - Update NE system `// Equation 12`
  - Compute $Task\_Novel$ `// Equation 8`
  - **if** $Task\_Novel = True$ **then**
    - K ← K+1
    - $task \leftarrow$ K
    - $Task\_Change \leftarrow$ False
    - create a new policy and reward predictor for the novel task
    - reset neuromodulation system

stand, walker-walk and walker-run. In the walker-stand task, the reward is a combination of terms encouraging an upright torso and some minimal torso height. The walker-walk and walker-run tasks include a component encouraging forward velocity. We list the hyperparameters in two experiments in Table 4.1.

Hyperparameters	GridWorld	Worker Robot
ch_{init}	0.5	0.5
ch_{max}	1.0	1.0
ch_{min}	0.1	0.1
ch_{change}	0.2	0.2
$ch_{expected}$	1.1	1.1
$ch_{unexpected}$	0.9	0.9
p_{match}	0.5	0.5
β	2	2
ne_{max}	1.0	1.0
ne_{init}	0.1	0.1
$ne_{unmatched}$	1.05	1.05
$ne_{threshold}$	0.9	0.9

Table 4.1: Hyperparameters of ACh and NE neuromodulatory systems.

In our experiments, the environment change is demonstrated as a task switch. For Gridworld, we set the task sequence for the experiment of Gridworld as [pickup-green, pickup-blue, pickup-yellow, pickup-green, pickup-blue, pickup-yellow]. For the MuJoCo robot, we set the task sequence for the walker as [walker-stand, walker-walk, walker-run, walker-stand, walker-walk, walker-run]. The task switch in each sequence requires the agent to quickly detect environment changes while the recurrence of tasks requires the agent to achieve successful task match and utilize prior learned knowledge.

Our neuromodulatory system is compatible with different types of reinforcement learning algorithms and settings. In the Gridworld experiments, the agent has a discrete action space and we use the Proximal Policy Optimization (PPO) algorithm Schulman et al. [2017], which is an on-policy RL method. In the MuJoCo walker robot experiment, the agent has a continuous action space and we use the Twin Delayed Deep Deterministic policy gradient (TD3) Fujimoto et al. [2018], which is an off-policy RL method.

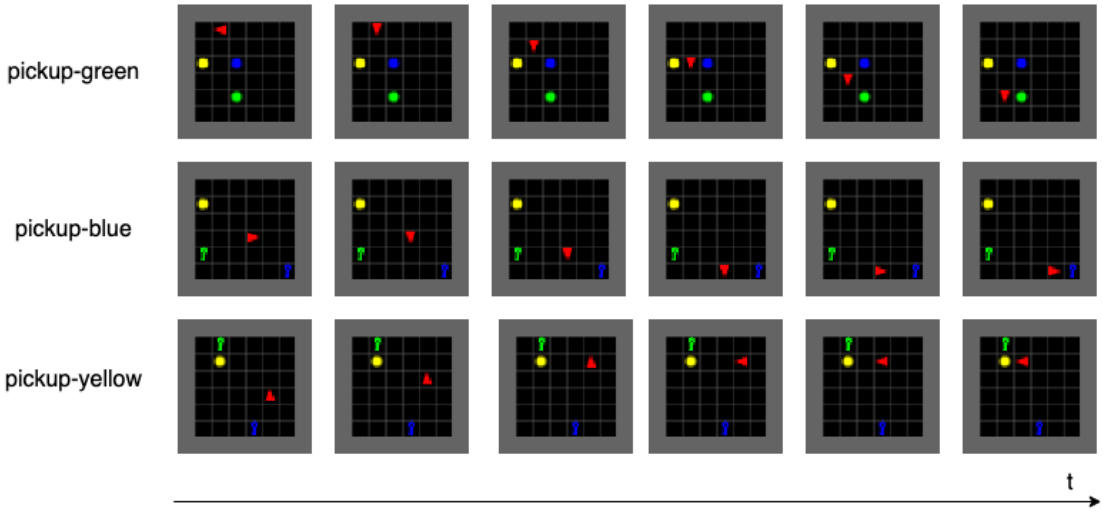


Figure 4.1: Gridworld environment. Examples of three tasks (pickup-green, pickup-blue and pickup-yellow) in Gridworld environment. In each task, the agent moves towards the object with a specified color.

Figure 4.2: MuJoCo environment. Examples of three tasks (stand, walk and run) of MuJoCo walker robot. The agent has unique behavior in each task.

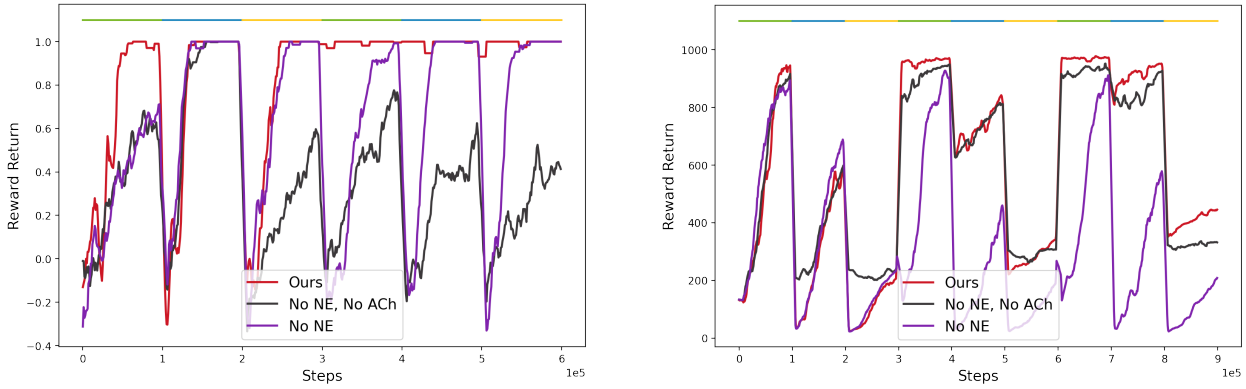
4.5 Results

4.5.1 Reinforcement Learning Performance

In both experiments, the addition of our neuromodulatory system improved performance. To demonstrate the efficacy of our method, we conducted an ablation study by disabling the NE system or disabling both the ACh and the NE system. Since the NE system depends on the ACh system, we cannot ablate ACh while keeping the NE system. We compared the performance of the RL agent with and without ablations (Figure 4.3). The better performance on novel tasks when neuromodulation was included due to the agent detecting environment changes quickly and avoiding detrimental knowledge transfer between tasks. Meanwhile, the immediate high score on recurring tasks demonstrates that the neuromodulatory system allows the agent to recognize previously observed tasks and utilize prior knowledge. In the ablation experiments, when the NE system is disabled, the agent cannot correctly detect a new task after a task change. As a result, all new tasks were simply recognized as novel tasks and the agent conducted learning from scratch every time a task change was detected. When the ACh system is also removed, the agent cannot detect task changes and conduct normal RL training throughout the whole task sequence. As a result, the agent would have only one task policy and the knowledge learned in the previous task will all be transferred to the next one. Its influence on the reward performance depends on the similarities of tasks in the task sequence. For example, the knowledge transfer in Gridworld is negative since the correct object to pick up in the last task will be wrong in the next task. As for tasks on walker robot, knowledge transfer could be more positive since learning to run could benefit from the knowledge of how to walk. However, in both two experiments, the RL agent with our neuromodulatory system achieved the best performance.

Besides the general RL performance, we're also interested in the benefits of our neuromodulatory system in fast performance recovery. In the experiments above, each task occurs

multiple times in the task sequence and the agent gradually improves performance via learning when interacting with each task. Although task changes regularly following the task sequence, an intelligent agent should quickly recover the performance when encountering the same task again. As a result, we compute the average time steps needed to recover 90% of the performance when facing a task that has been learned before. As shown in Table 4.2, our neuromodulatory system allows the agent to achieve much faster performance recovery compared to the ablated agents.



(a) RL Performance on Gridworld

(b) RL Performance on walker robot

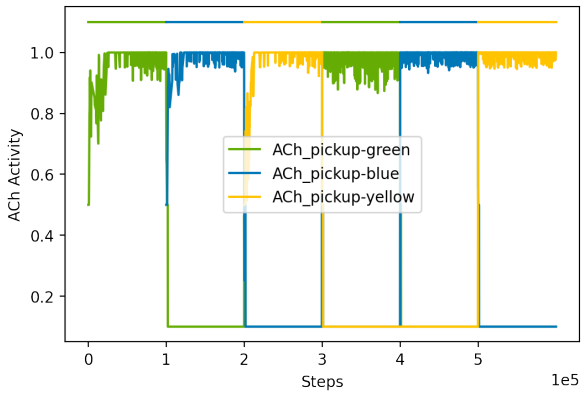
Figure 4.3: Results of reinforcement learning performance with and without the neuromodulatory system. The bar on top of the figure represents the task sequence. The task sequence in Gridworld experiment is [pickup-green, pickup-blue, pickup-yellow, pickup-green, pickup-blue, pickup-yellow] and the task sequence in walker robot experiment is ['walker-stand', 'walker-walk', 'walker-run', 'walker-stand', 'walker-walk', 'walker-run', 'walker-stand', 'walker-walk', 'walker-run'].

	Time Steps of Performance Recovery		
	Ours(Intact)	No NE	No NE, No ACh
pickup-green	**150.3**	45060.5	59753.3
pickup-blue	**130.7**	25510.7	69535.2
pickup-yellow	**143.3**	29428.7	87605.6
walker-stand	**796.0**	311000.0	354500.0
walker-walk	**1843.0**	411500.0	598000.0
walker-run	**1542.2**	501000.0	585333.3

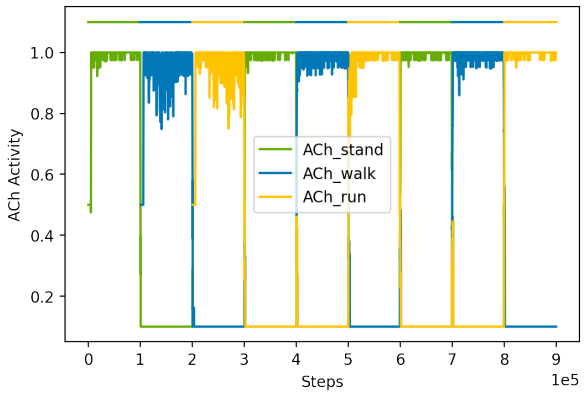
Table 4.2: Average time steps needed to achieve 90% performance recovery of each task of our method and ablated studies. The results are averaged over 6 runs.

4.5.2 Activity of Neuromodulatory System

We examined the activity of the ACh and NE systems in these two experiments. As shown in Figure 4.4, our ACh system is able to track the uncertainty on each task and always show high certainty on the correct task. This allows the agent to efficiently identify the correct task identity and adapt to task switches.



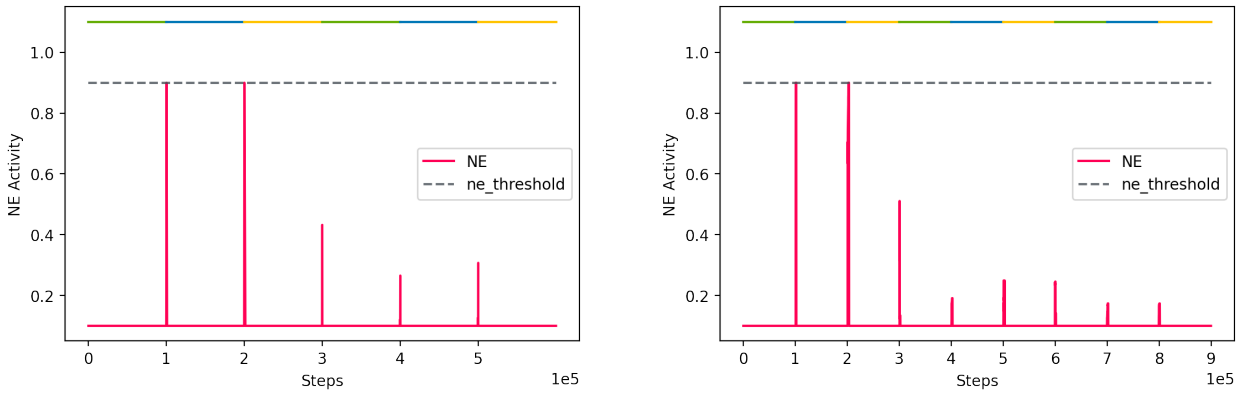
(a) ACh activity for the Gridworld experiment.



(b) ACh activity for the walker robot experiment.

Figure 4.4: Dynamics of the activity of the ACh system in the two experiments. The bar on top of the figure represents the task sequence.

As shown in Figure 4.5, our NE system shows high activity for the first and second task switches since the second and third tasks are novel. The activity of the NE system also increased for other task switches but didn't reach the threshold because those switched tasks have been encountered before by the agent and were not novel. This demonstrates that our NE system can track the unexpected uncertainty of the environment which allows the agent to distinguish between novel and familiar tasks. This capability helps the agent to better adapt to environment changes.



(a) NE activity for the Gridworld experiment.

(b) NE activity for the walker robot experiment

Figure 4.5: Dynamics of the activity of the NE system in the two experiments. The bar on top of the figure represents the task sequence.

4.6 Conclusion

We developed a system inspired by neuromodulation to track the uncertainty of the environment and help reinforcement learning agents to quickly adapt to environment changes. We demonstrated the efficacy of the neuromodulatory system in two reinforcement learning experiments; namely, a Gridworld environment and a simulated walker robot. We believe these results provide insights into how intelligent agents survive in uncertain environments and also enable the deployment of artificial agents in complicated real-world applications.

Chapter 5

Domain Adaptation in Reinforcement Learning via Latent Unified State Representation

5.1 Introduction

Deep reinforcement learning has been successful in a series of control problems, such as Atari 2600 video games [Mnih et al., 2013] and MuJoCo environments [Lillicrap et al., 2015]. However, the advances of deep RL relies on a large amount of interactions with the environment. In addition, the policy tends to specialize to the training domain and fails

to generalize to new domains even when these two domains are similar. It has been shown that slight visual changes on pixel-based observations from Atari games could cause the well trained policy totally break down [Gamrian and Goldberg, 2019]. These two limitations make deep reinforcement learning algorithms inefficient when applied to sets of tasks. As a result, efficient domain adaptation approaches are important for the applicability of Deep RL.

Although state-of-the-art methods have demonstrated compelling performance in domain adaptation in RL, these approaches all have their limitations. Domain randomization [Tobin et al., 2017, Andrychowicz et al., 2020, Slaoui et al., 2020] relies on the availability of multiple source domains for training and cannot be applied in one-to-many generalization scenarios. Image-to-image translation approaches [Pan et al., 2017, Tzeng et al., 2020, Gamrian and Goldberg, 2019] need a computationally expensive generator model for image translation. The extra burden on computation brought by the generator model is impractical for real-time applications such as autonomous driving. Other approaches utilize the latent embedding of encoder-decoder models to extract internal state representation for better generalization Higgins et al. [2017]. However, domain-specific variations are also compressed into the latent embedding which could be problematic for zero-shot policy transfer.

To solve the problem of domain adaptation across related RL tasks and avoid limitations of prior methods, we propose to learn a latent unified state representation (LUSR) for different domains and then train RL agents in the source domain based on that. After the RL training, zero-shot policy transfer is evaluated in target domains. To learn LUSR, we split the latent state representation into domain-general embedding which contains information existing in all domains and domain-specific embedding that compress domain specific information. LUSR is composed of domain-general embedding only and thus is able to ignore domain-specific variations and generalize across domains.

To empirically justify our approach, we conducted experiments in two car driving tasks

with different visual complexity. We first applied our approach in CarRacing games with analysis of final domain adaptation performance, domain adaptation performance across the training period, generalization to totally unseen domains and policy explanation with saliency maps. Then we evaluated our approach in autonomous driving tasks in CARLA simulator [Dosovitskiy et al., 2017a] with more challenging and realistic visual observations.

In comparison with other approaches, LUSR does not need RL training in multiple source domains like domain randomization and thus is applicable to a wider range of tasks. In addition, LUSR does not need computationally expensive generator models and can achieve better training efficiency compared with image-to-image translation approaches that operate in pixel-space. Finally, in contrast with other approaches that use latent state representation, LUSR filters out the factors of variation across domains and ensures the latent state representation is unified across all domains.

5.2 Related Work

Related work either tried to tackle domain adaptation in RL by directly generalizing the policy or learning generalized state representations.

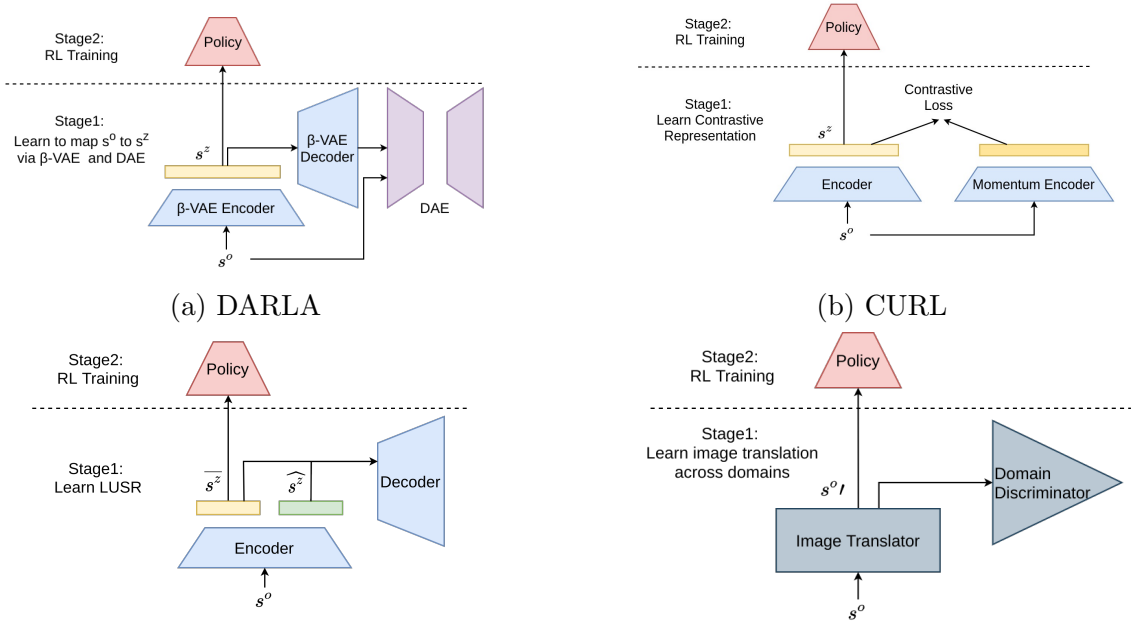
Domain randomization is the most popular approach to directly learn a policy with generalization capability [Tobin et al., 2017, Andrychowicz et al., 2020, Slaoui et al., 2020, Laskin et al., 2020a]. By training on many source domains, the RL agent learns to ignore irrelevant factors of variation and attend to common features only. However, this approach relies on the availability of multiple source domains for training and the complexity of this approach scales with the number of variations.

Instead of learning a policy with generalization capability directly, other works focus on the generalization of state representations. Some visual domain adaptation works use image-to-

image translation to map the pixel-based states in the target domain to the paired states in the source domain [Pan et al., 2017, Tzeng et al., 2020, Gamrian and Goldberg, 2019]. This is generally achieved via adversarial methods such as Generative Adversarial Networks (GANs) [Goodfellow et al., 2014], and Unaligned GANs [Liu et al., 2017, Zhu et al., 2017] in the case where image pairs are lacking. While these methods provide promising results, the image translation brings extra burden during inference time which is impractical in real-time applications.

Other works take one step further and try to learn a generalized state representation by mapping pixel-based states to a latent space [Higgins et al., 2017]. For example, the latent embedding of variational autoencoder (VAE) can be used as an internal latent state representation in RL. We call this method as VAE-Embedding. DARLA further extends the VAE to β-VAE to encourage the disentanglement of the latent embedding and uses one internal layer of a pre-trained Denoising AutoEncoder (DAE) [Vincent et al., 2010] as the reconstruction target. Although disentanglement in latent state representation makes it easier for RL agents to ignore irrelevant domain-specific features, the policy transfer performance is not guaranteed because domain-specific features still reside in the latent state representation and their contribution to the policy output cannot be generalized to other domains. CURL extracts high-level features from raw pixels using contrastive learning and greatly improves the sample efficiency [Laskin et al., 2020b].

In this work, we choose VAE-Embedding, DARLA, CURL and CycleGAN-based image-to-image translation as benchmarks. To make it more clear how LUSR differs from them, we use Figure 5.1 to demonstrate their frameworks.



(a) DARLA

(b) CURL

(c) LUSR (ours)

(d) CycleGAN Image Translation

Figure 5.1: Architectures of our method (LUSR) and other benchmarks (DARLA, CURL and CycleGAN based image-to-image translation) used in this work for comparison. The architecture of VAE-Embedding could be considered as a special case of DARLA that replaces β-VAE with VAE and avoids the usage of DAE. The learning of all these approaches could be divided into two stages. The first stage is learning appropriate state representations that support domain adaptation in RL and the second stage is doing RL training.

5.3 Domain Adaptation in Reinforcement Learning

Reinforcement learning is an area that studies how agents should take actions in an environment in order to maximize their cumulative rewards. The environment is typically stated in the form of a Markov decision process (MDP), which is expressed in terms of the tuple (S, A, T, R) where S is the state space, A is the action space, T is the transition function and R is the reward function. At each time step t in the MDP, the agent takes an action a_t in the environment based on current state s_t and receives a reward r_{t+1} and next state s_{t+1}. The goal of the agent is to find a policy $\pi(s)$ to choose actions that maximize the discounted cumulative future rewards $r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...$, where γ is the discount factor ranging from 0 to 1.

To formalize domain adaptation scenarios in the setting of reinforcement learning, we define the source and target domains as D_S and D_T. Each domain corresponds to a MDP defined as tuple (S, A, T, R) and thus the MDPs in the source domain D_S and target domain D_T are defined as (S_S, A_S, T_S, R_S) and (S_T, A_T, T_T, R_T), respectively. The source and target domains could have distinct state spaces S, but their action spaces A should be the same and their transition function T and reward function R should have similarity because of the sharing internal dynamics. Namely, we focus on policy transfers where $T_S \approx T_T$, $R_S \approx R_T$, $A_S = A_T$, but $S_S \neq S_T$.

Take autonomous driving as an example, different domains may correspond to different weather conditions. For instance, the source domain is driving on a sunny day and the target domain is driving on a rainy day. While state space S (visual observations) could differ due to rain and different lighting conditions, the action space A (throttle and steering) remains the same. As for the transition function T and reward function R, they should have similarity since the state transition for both domains are governed by the traffic condition and driving control while the reward function for both domains are determined by the movement of the vehicle.

5.4 Methods

Our approach focuses on learning a latent unified state representation (LUSR) for states from different domains in RL. In this section, we first introduce the definition of LUSR and then introduce how to learn it.

5.4.1 LUSR Definition

We first introduce two notions for state space in RL which are the agent's raw observation state space S^o and the agent's internal latent state space S^z. Raw observation states s^o consists of a grid of pixels while each unit in the internal latent state s^z represents a high level semantic feature. A mapping function $\mathcal{F} : S^o \to S^z$ maps the observation state to the corresponding internal latent state. In our work, high level semantic features in S^z are further divided into domain-specific ones (such as weather conditions in the driving task) and domain-general ones (such as vehicle dynamics). Here we denote $S^z = (\widehat{S^z}, \overline{S^z})$ where $\widehat{S^z}$ represents domain-specific features and $\overline{S^z}$ represents domain-general features. For state representation in source and target domains, this is summarized as

$$\begin{aligned} S_S^o &\neq S_T^o \\ S_S^z = (\widehat{S_S^z}, \overline{S_S^z}); \quad S_T^z &= (\widehat{S_T^z}, \overline{S_T^z}) \\ \overline{S_S^z} = \overline{S_T^z}; \quad \widehat{S_S^z} &\neq \widehat{S_T^z} \end{aligned} \tag{5.1}$$

In our setting of domain adaptation, the transition function T and reward function R only depend on $\overline{S^z}$ which is consistent across domains. Here we define the reward and transition function that take s^o as input as R^o and T^o while the reward and transition function that take s^z as input as R^z and T^z. Then, we have

$$\begin{aligned} T_S^o \neq T_T^o; \quad R_S^o &\neq R_T^o \\ T_S^z = T(\overline{S_S^z}) = T(\overline{S_T^z}) &= T_T^z \\ R_S^z = R(\overline{S_S^z}) = R(\overline{S_T^z}) &= R_T^z \end{aligned} \tag{5.2}$$

Since $\overline{S^z}$ is consistent across domains and the reward structure (T and R) depend only on this representation (not on $\widehat{S^z}$), the RL agent taking $\overline{S^z}$ as input will be able to be trained successfully and the trained agent also has the capability to adapt from the source domain to target domains. As a result, the goal of our approach is learning the mapping function $\mathcal{F} : S^o \rightarrow \overline{S^z}$ that maps raw observation states to the latent unified state representation which we call LUSR.

5.4.2 Learning LUSR

In this work, we choose to learn the mapping function $\mathcal{F} : S^o \rightarrow \overline{S^z}$ via Cycle-Consistent VAE [Jha et al., 2018] which is a non-adversarial approach to disentangle domain-general and domain-specific factors of variation. Similar to VAE [Kingma and Welling, 2013], Cycle-Consistent VAE is also composed of an encoder and a decoder. However, the output from the encoder is split into domain-general and domain-specific embeddings. To learn the mapping function $\mathcal{F}$, a number of random observation states from a set of pre-defined domains are first collected and then used as input for Cycle-Consistent VAE model training. Once the model is trained, the encoder is able to map observation states s^o from any domain in the domain set to a latent state representation composed of $\overline{s^z}$ and $\widehat{s^z}$. As a result, we use the trained encoder as our mapping function $\mathcal{F}$ and keep only domain-general representation as LUSR.

Cycle-Consistent VAE is based on the idea of cycle consistency whose intuition is that two well trained forward and reverse transformations composed together *in any order* should approximate an identity function. For example, in the VAE, the encoder is a forward transformation that converts an input image to a latent vector while the decoder is the reverse transformation that converts the latent vector back to a reconstructed image. Here we define the forward cycle as: $Dec(Enc(s^o)) = s^{o\prime}$ and the reverse cycle as $Enc(Dec(\widehat{s^z}, \overline{s^z})) = (\widehat{s^{z\prime}}, \overline{s^{z\prime}})$.

As indicated by the cycle consistency, $s^{o\prime}$ should be close to s^o and also $(\widehat{s^{z\prime}}, \overline{s^{z\prime}})$ should be close to $(\widehat{s^z}, \overline{s^z})$.

In the forward cycle of Cycle-Consistent VAE, for two observation states s_1^o, s_2^o from the same domain, $Enc(s_1^o) = \widehat{s_1^z}, \overline{s_1^z}$ and $Enc(s_2^o) = \widehat{s_2^z}, \overline{s_2^z}$. Since both originate from the same domain and $\widehat{s^z}$ contains only domain-specific information, swapping $\widehat{s_1^z}$ and $\widehat{s_2^z}$ should have no effect on the reconstruction loss which means we should get $Dec(\widehat{s_2^z}, \overline{s_1^z}) \approx s_1^o$ and $Dec(\widehat{s_1^z}, \overline{s_2^z}) \approx s_2^o$. This operation ensures that domain-specific information and domain-general information are compressed into $\widehat{s^z}$ and $\overline{s^z}$ separately.

In the reverse cycle, a randomly sampled $\overline{s^z}$ is passed through the decoder in combination with two domain-specific embeddings $\widehat{s_1^z}$ and $\widehat{s_2^z}$ to obtain two reconstructed images $s_1^{o\prime}$ and $s_2^{o\prime}$. Since both $s_1^{o\prime}$ and $s_2^{o\prime}$ are generated based on the same $\overline{s^z}$, their corresponding domain-general latent embedding $\overline{s_1^z}\prime$ and $\overline{s_2^z}\prime$ should also be the same.

As a result, the objective for Cycle-Consistent VAE to minimize is

$$\mathcal{L}_{cyclic} = \mathcal{L}_{forward} + \mathcal{L}_{reverse} \tag{5.3}$$

where

$$\begin{aligned}
\mathcal{L}_{forward} =& - \mathbb{E}_{q_\phi(\overline{s^z}, \widehat{s^z}|s^o)}[\log p_\theta(s^o|\overline{s^z}, \widehat{s^z}*)] \\
&+ KL(q_\phi(\overline{s^z}|s^o)||p(\overline{s^z})) \\
\mathcal{L}_{reverse} =& \mathbb{E}_{\overline{s^z}\sim p(\overline{s^z})}[||\overline{q_\phi}(p_\theta(\overline{s^z}, \widehat{s_1^z})) - \overline{q_\phi}(p_\theta(\overline{s^z}, \widehat{s_2^z}))||_1]
\end{aligned}$$

$\mathcal{L}_{forward}$ here is a modified variational upper-bound and $\mathcal{L}_{reverse}$ is the loss for cycle consistency. q_ϕ and p_θ are parameterized functions of the encoder and decoder. We define $\overline{q_\phi}$ as q_ϕ that only keeps the domain general embedding as output. The latent embedding s^z is composed of $\overline{s^z}$ and $\widehat{s^z}$ which are domain-general and domain-specific latent embeddings

corresponding to observation state s^o. $\widehat{s^z*}$ represents any random domain-specific embedding from the same domain while $\widehat{s_1^z}$ and $\widehat{s_2^z}$ are two different domain-specific embeddings.

5.5 Experiments

We first apply our approach to a set of CarRacing variants which allows manual manipulations of the visual observations. This flexibility allows us to analyze the influence of different categories of variations on the performance of domain adaptation in RL. After that, our approach is applied in autonomous driving tasks in the CARLA simulator in which the observational states are much more complicated and helps us to evaluate the ability of our approach to scale up to more challenging tasks.

5.5.1 CarRacing

We first apply our approach on variants of CarRacing game which is a continuous control task to learn to drive from pixels. As shown in Figure 5.2, we divide all variants into three categories: source domain, seen target domains and unseen target domains. We first collect random observation states from the source domain and seen target domains to learn the mapping function $\mathcal{F}$ which maps raw observation states to LUSR. In each domain, we collect 100k images and thus have 500k images in total (one source domain and four seen domains). The collected images are used as the dataset to train a Cycle-Consistent VAE model whose encoder is the mapping function $\mathcal{F}$ we need. After that, we train the RL agent in the source domain with LUSR for 10 millions steps via Proximal Policy Optimization (PPO) [Schulman et al., 2017] algorithm. In this work, we use Ray RLlib [Liang et al., 2018] and RLCodebase [Xing, 2020] for the PPO implementation. After the RL training, we test the RL agent's performance of adapting to the seen target domains and unseen target

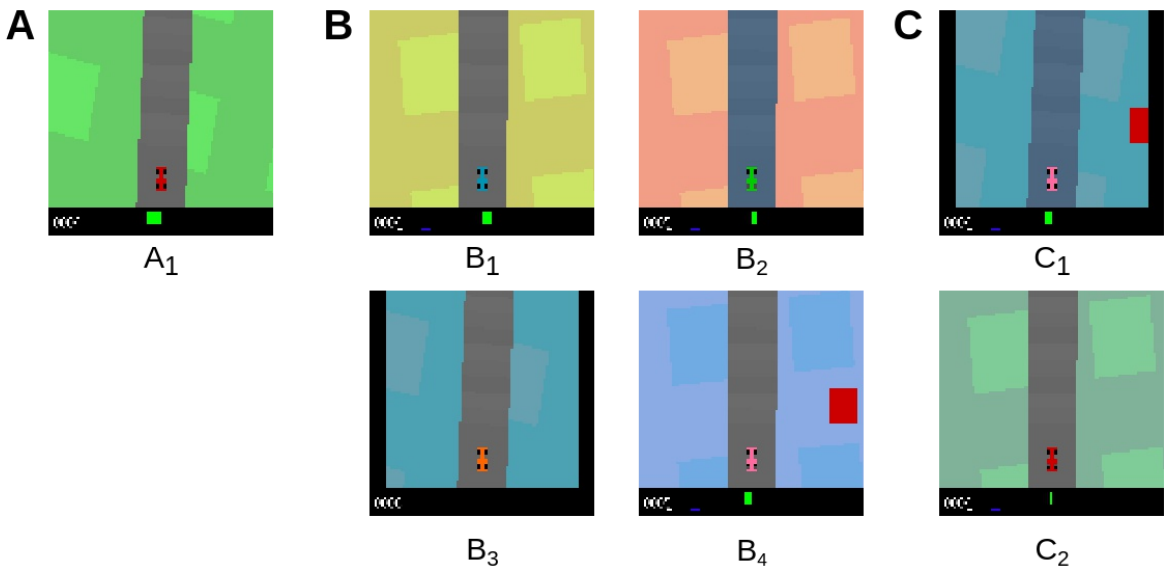


Figure 5.2: Variants of CarRacing games. **A**. The original version of CarRacing game which is set as the source domain. **B**. The seen target domains of CarRacing games whose observation states are collected for learning LUSR. **C**. The unseen target domains of CarRacing games. These two domains are never exposed to the agent, not only during RL training but also during latent state representation learning.

domains.

With the ability of inducing manual manipulations over the observation states, we design two types of variations. The first type is color change including changing the background color, the car color and the road color. For example, the background color in all target domains is different from that in the source domain. Another type of variation is inducing patterns. For example, we induce a red blob at a fixed position in the fourth game of seen target domains (B_4). In summary, compared with the source domain A_1, seen target domains B_1 and B_2 have color changes while B_3 and B_4 have both color changes and new patterns. For unseen target domains, C_1 combines all variations introduced in seen target domains and C_2 uses a totally new background color.

Figure 5.3: Experiment of the driving task in CARLA simulator. Examples of the driver view (observation states) under three different weather conditions: evening, clear noon and hard rain from left to right.

5.5.2 Autonomous Driving in CARLA

Although CarRacing games are suitable to study the domain adaptation problem of RL agents, the observation states are relatively simple compared to real world observations during driving. To further evaluate the performance of our approach, we applied it in a much more challenging task: autonomous driving in the CARLA simulator. In this experiment, we first choose a start point and an end point in the map of town07 for the driving task. To go from the start point to the destination, the vehicle must go through a curvy road and avoid collisions and lane crossings. The action space is composed of two continuous values for driving control (throttle and steering). At each step, the driving control is applied on the vehicle for 0.1 simulation second. We use images captured by a camera attached to the front end of the RL agent vehicle along with the current speed as the observation states. Each episode terminates if the vehicle collides, runs out of the lane, reaches the destination, or reaches the maximum episode timesteps (800 in this experiment). To make the CARLA simulator compatible with RL training, we use a gym wrapper of CARLA in the experiment [Chen et al., 2019].

To study domain adaptation, we test the model under different weather conditions and times

of day. Specifically, the RL agent is first trained in the late evening and then tested in the weather of clear noon and hard rain. Examples of the driver view under these conditions are shown in Figure 5.3.

Since CARLA aims to provide realistic simulations of urban driving, the observation states in this driving task are much more complex and challenging for domain adaptation compared to states in CarRacing games. Besides that, the complexity of environment dynamics also makes the simulation of CARLA slower compared to CarRacing games. As a result, we set the number of PPO training steps in this experiment as 50k. This further requires the RL agent to have a high training efficiency to achieve good performance with limited number of interactions with the environment.

5.6 Results and Discussion

In this section, we introduce the results of our approach in two experiments along with other benchmarks.

5.6.1 CarRacing

LUSR Demonstration

We first demonstrate the effectiveness of LUSR. In our approach, the latent embedding is split into domain-general embedding $\overline{s^z}$ and domain-specific embedding $\widehat{s^z}$. To verify that these two embeddings are well disentangled, we first select random images from the source domain and seen target domains and then extract their latent embeddings. For example, we get $\widehat{s_1^z}$ and $\overline{s_1^z}$ for image s_1^o, and $\widehat{s_2^z}$ and $\overline{s_2^z}$ for image s_2^o. If we feed the decoder with a latent embedding composed of $\widehat{s_1^z}$ and $\overline{s_2^z}$, the reconstructed image $s^{o\prime}$ should have visual

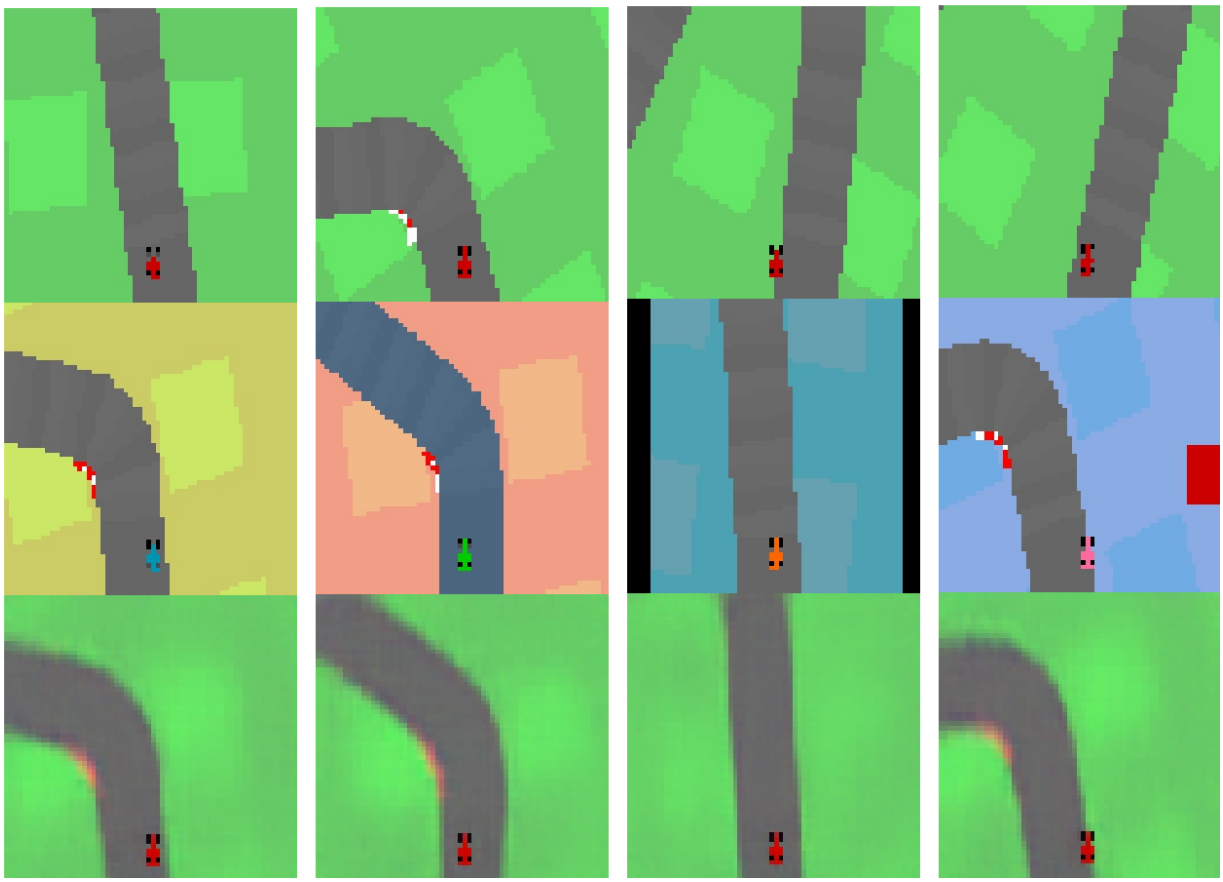


Figure 5.4: Results of Cycle-Consistent VAE. The first row are four random images from the source domain and the second row are four random images from four seen target domains respectively. The last row are reconstructed images that take $\widehat{s^z}$ from the first row and $\overline{s^z}$ from the second row.

features from both s_1^o and $_2^o$. Furthermore, the shared features between $s^{o\prime}$ and s_1^o should be domain-specific while the shared features between $s^{o\prime}$ and s_2^o should be domain-general. As shown in Figure 5.4, the third row of images are generated with $\widehat{s^z}$ from the first row and $\overline{s^z}$ from the second row. As a result, their domain-specific features (color and patterns) are the same as the first row of images while the domain-general features (road shape) are the same as images in the second row.

Domain Adaptation After Training

After the RL training in the source domain, we evaluate the domain adaptation performance of our approach and other benchmarks in both seen target domains and unseen target domains (see table 5.1). The result shows that RL agents trained with LUSR are able to generalize to all target domains almost without performance loss and achieve best scores in most target domains. For DARLA, the choice of parameter β strongly affects the adaptation performance. Furthermore, it generalizes better in target domains with only color variations and could fail to adapt to domains with new patterns. VAE-Embedding has notable performance loss for all target domains. CURL has the worst transfer performance among all approaches and completely fails in most domains. Finally, CycleGAN can also adapt to all target domains without performance loss. However, the final scores are not comparable to other approaches that use latent embeddings as input for RL training.

Domain Adaptation During Training

Besides the domain adaptation performance after training, we're also interested in the adaptation performance during the training period. Since the RL agent will be more and more deterministic in action selection during the training, the domain adaptation performance could also be affected. As a result, we evaluate the model adaptation performance every

Approach	Source Domain	Seen Target Domains				Unseen Target Domains	
	CarRacing_A1 Score	CarRacing_B1 Score(Ratio)	CarRacing_B2 Score(Ratio)	CarRacing_B3 Score(Ratio)	CarRacing_B4 Score(Ratio)	CarRacing_C1 Score(Ratio)	CarRfacing_C2 Score(Ratio)
LUSR	805.13	803.52 (**1.00**)	**807.37** (**1.00**)	**803.11** (**1.00**)	**781.94** (**0.97**)	**678.7** (**0.84**)	**800.56** (**0.99**)
DARLA($\beta = 10$)	845.87	645.81 (0.76)	250.85 (0.30)	-72.99 (-0.09)	-62.96 (-0.07)	-65.72 (-0.08)	631.09 (0.75)
DARLA($\beta = 30$)	851.48	**834.99** (**0.98**)	**819.05** (**0.96**)	-60.76 (-0.07)	-73.18 (-0.09)	-72.55 (-0.09)	**806.76** (**0.95**)
DARLA($\beta = 100$)	778.78	704.27 (0.90)	207.9 (0.27)	451.81 (0.58)	27.77 (0.04)	182.35 (0.23)	539.63 (0.69)
VAE-Embedding	816.74	616.89 (0.76)	282.71 (0.35)	484.57 (0.59)	223.88 (0.27)	332.58 (0.41)	595.42 (0.73)
CURL	748.58	560.23(0.75)	-44.24(-0.06)	-55.29(-0.07)	-32.45(-0.04)	-113.13(-0.15)	-69.23(-0.09)
CycleGAN	709.12	707.64 (**1.00**)	704.33 (**0.99**)	713.86 (**1.01**)	711.85 (**1.00**)	**715.43**(**1.01**)	671.67(**0.96**)

Table 5.1: Domain adaptation performance of LUSR and benchmarks in CarRacing games. We train 3 models for each approach and evaluate each model for 100 episodes in each domain after training. The average final score of 3 models are reported in the table for each approach. We also report the ratio of scores achieved in target domains to the score achieved in the source domain to demonstrate the policy transfer performance.

Approach	Source Domain		Target Domains			
	CARLA (Evening)		CARLA (Clear Noon)		CARLA (Hard Rain)	
	Score	Steps	Score	Steps	Score	Steps
LUSR	**1125.06**	469.1	**1175.61**	565.3	**1270.32**	515.6
DARLA	841.59	342.0	194.41	134.2	187.97	119.9
VAE-Embedding	1113.90	384.9	674.42	744.2	846.14	527.4
CURL	1112.42	521.2	44.34	42.4	73.63	60.2
CycleGAN	333.57	175.1	333.88	174.9	332.71	163.6

Table 5.2: Domain adaptation performance of LUSR and benchmarks in CARLA autonomous driving tasks. We train 3 models for each approach and choose the best model for evaluation. Each model is evaluated for 10 episodes. The average score and time steps spent in each episode are reported in the table.

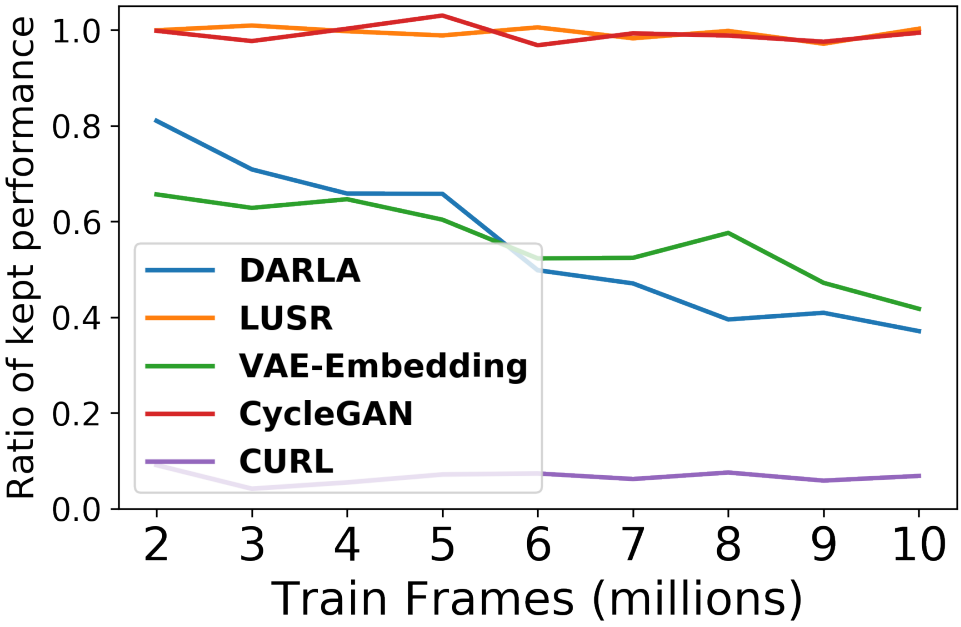


Figure 5.5: Domain adaptation performance during training in CarRacing comparing LUSR to other benchmarks.

1 million frames of training for all approaches. Our result shows that both LUSR and CycleGAN have consistent adaptation performance during the whole training period while the adaptation performance of DARLA and VAE-Embedding gradually decrease (see Figure 5.5). This demonstrates that domain-specific features do contribute to the RL policy output if they are included in the latent state representation and their influence will be more and more problematic as RL training goes on.

Saliency Map

Saliency map is an approach to visualize and understand the behavior of RL agents. In this work, we also use saliency maps [Greydanus et al., 2018] to visualize how RL agents trained with different methods attend to the observation states (see Figure 5.6). The result shows that RL agent trained with LUSR has more centralized attention and mainly attends to the center of the road. In comparison, the saliency maps generated by other approaches are

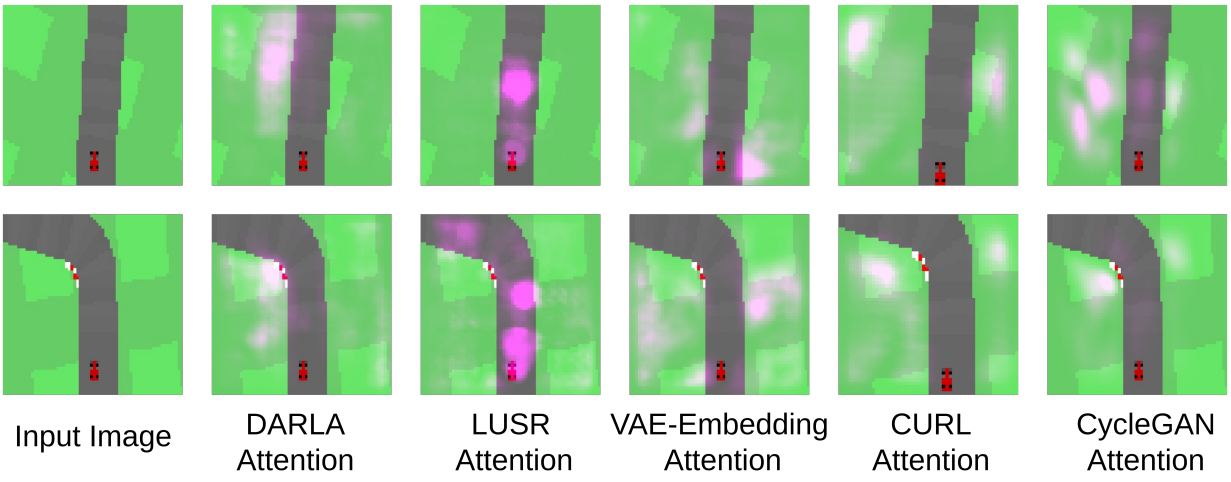


Figure 5.6: Examples of saliency maps generated by RL agents trained via DARLA, LUSR, VAE-Embedding, CURL and CycleGAN. The RL agent trained with LUSR has the most centralized attention and mainly attends to the center of the road.

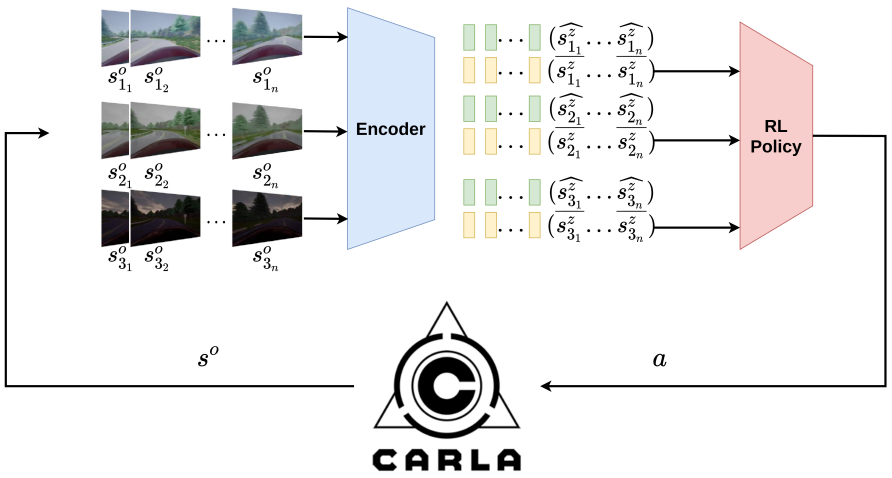
much more diffused and attend more to the edges between road and grass rather than road itself. Although it also makes sense to learn to drive based on the edges, the contrast between them could also change when the color of grass changes and thus brings more challenges in generalization. This may explain why LUSR has better generalization performance than other benchmarks.

5.6.2 Autonomous Driving in CARLA

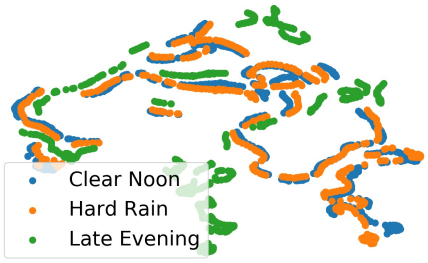
We further apply our approach in the autonomous driving task in CARLA simulator whose observation states are more complicated and thus increase the difficulty of RL training and generalization.

LUSR Demonstration

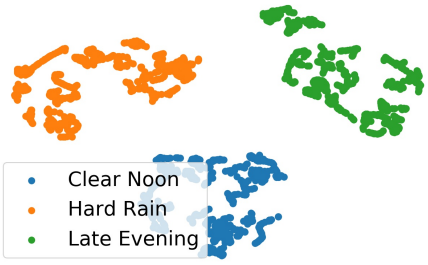
We also demonstrate the disentanglement of domain-specific embedding and domain-general embedding for images in CARLA simulator (see Figure 5.7). We first collect paired ob-



(a)



(b)



(c)

Figure 5.7: Demonstration of the disentanglement of domain-general embedding and domain-specific embedding in related CARLA driving tasks. **a**. The workflow of generating paired observational states and extracting latent embeddings. **b**,**c**. t-SNE plot of the domain-general and domain-specific embeddings from three CARLA driving tasks.

servation images from three tasks by placing the vehicle at the same starting point in the map and taking same actions in the driving. After collecting images, we extract their latent domain-general and domain-specific embeddings via a trained encoder and show their t-SNE plots in Figure 5.7b and 5.7c. It shows that domain-general embeddings from different tasks do have close similarities while domain-specific embeddings from different tasks are well clustered separately. The result demonstrates the disentanglement of domain-general and domain specific embeddings for images in CARLA driving tasks and thus supports the feasibility of LUSR in more challenging scenarios.

Domain Adaptation Performance

For domain adaptation in CARLA autonomous driving tasks (see Table 5.2), RL agents trained with LUSR is able to achieve zero-shot policy transfer without performance loss for both two target domains. It also achieves best scores in all domains compared with other approaches which shows the training efficiency of LUSR.

Different from in CarRacing games, DARLA fails to adapt the trained policy in two target domains in CARLA autonomous driving tasks. This may be due to DARLA increasing the disentanglement of the latent embedding at the sacrifice of information accuracy. For complicated observation states like images in CARLA, it's much more difficult to achieve good disentanglement of each latent unit and the problem of information loss in the latent embedding is more serious. This causes the quality of the latent embedding in DARLA to be worse than LUSR and VAE-Embedding. This argument is supported by the results that higher β in DARLA leads to worse RL training performance.

VAE-Embedding achieves similar training performance in the source domain as LUSR while its adaptation performance in target domains is worse. Besides that, its driving behavior in target domains is very different from the behavior in the source domain. When adapting to target domains, the RL agent drives much slower and frequently reaches the time limit of each episode. As shown in table 5.2, compared to the result in the source domain, RL agents trained with VAE-Embedding receive lower scores while spending more time steps in each episode when driving in target domains.

Although CURL achieves comparable training performance in the source domain as other approaches, it completely fails to generalize to the two target domains in CARLA. To understand the reason, we conducted a tSNE analysis for CURL. The result reveals clusters based on domain labels. We believe the reason is that domain specific features are very useful in learning to assign low similarities for two states from different domains during CURL

training and thus much domain specific information resides in the embedding of CURL. This prevents generalization across domains.

CycleGAN is also able to generalize to target domains well. However, it relies on pixel-wise input and the training efficiency is limited compared with other approaches that utilize internal latent state representation.

5.7 Conclusion

In this work, we propose to disentangle domain-general embedding and domain-specific embedding in the latent state representation of RL and theoretically formalize it in the scenario of domain adaptation. We propose LUSR which utilizes the domain-general latent embedding as state representation and prove its efficiency in two RL tasks with different visual complexity. As a result, our work enhances the applicability of Deep RL to real-world tasks that need both good domain adaptation performance and high training efficiency.

Chapter 6

Achieving Efficient Interpretability of Reinforcement Learning via Policy Distillation and Selective Input Gradient Regularization

6.1 Introduction

Reinforcement learning (RL) systems have achieved impressive performance in a wide range of simulated domains such as games [Mnih et al., 2015, Silver et al., 2016, Vinyals et al.,

2019, Wang et al., 2020], robotics [Lillicrap et al., 2015, Fujimoto et al., 2018, Haarnoja et al., 2018], automatic control [Li et al., 2017, Wang et al., 2022] and computer vision tasks [Le et al., 2021]. However, the interpretability of an agent's decision making and robustness to attacks need to be addressed when applying RL to real-world problems. For instance, in a self-driving scenario, real-time interpretability could explain how an RL agent produces a decision in response to its observed states and enable a safer deployment under real-world conditions and adversarial attacks [Ferdowsi et al., 2018, McAllister et al., 2019, Bojarski et al., 2018, Chen et al., 2020b].

Saliency maps in deep learning are used to interpret input features that are believed to be important for the neural network output [Simonyan et al., 2013, Selvaraju et al., 2017, Fong and Vedaldi, 2017, Smilkov et al., 2017, Sundararajan et al., 2017, Zhang et al., 2018, Nguyen et al., 2019]. As the issue of interpretability in RL gets more attention, a number of methods have been proposed to generate saliency maps to explain the decision making of RL agents. Existing saliency map methods in RL either used gradients to estimate the influence of input features on the output [Wang et al., 2016] (gradient-based methods) or computed the saliency of an input feature by perturbing it and observing the change in output [Greydanus et al., 2018, Iyer et al., 2018, Puri et al., 2020] (perturbation-based methods). Gradient-based methods can compute saliency maps efficiently with backpropagation. However, the quality of these gradient-based saliency maps was generally poor [Rosynski et al., 2020]. Perturbation-based methods are effective in highlighting the important features of the input, but at a significant computational cost, which can make them ineffective when deployed on systems with real-time constraints. As a result, existing RL agents cannot provide high interpretability in a computation-efficient manner.

Different from previous work proposing new saliency calculation methods, we focus on improving the natural interpretability of RL policies. Given an RL policy, we propose an approach of Distillation with selective Input Gradient Regularization (DIGR) that uses pol-

icy distillation and input gradient regularization to retrain a new policy. In our approach, input gradient regularization selectively regularizes gradient-based saliency maps of the policy to imitate its interpretable perturbation-based saliency maps. This allows the new RL policy to generate high-quality saliency maps with gradient-based methods and thus achieve both high interpretability and computational efficiency. At the same time, to ensure that input gradient regularization does not cause task performance degradation, we use policy distillation [Czarnecki et al., 2019] to constrain the output of the new RL policy to remain close to the original RL policy.

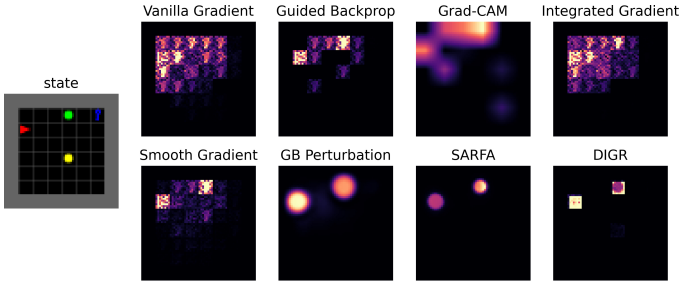
We evaluate our method in three different tasks, which include an object fetching task from MiniGrid [Chevalier-Boisvert et al., 2018], Breakout from Atari games and CARLA Autonomous Driving [Dosovitskiy et al., 2017b]. The results show that RL policies trained with our approach are able to achieve efficient interpretability while maintaining good task performance. Selective input gradient regularization also improves the robustness of RL policies to adversarial attacks. These two desired properties allow the RL policy to better adapt to real-world scenarios.

To summarize, we demonstrate a novel approach to improve the efficient interpretability and robustness on attacks on RL policies based on the utilization of saliency maps. Our approach increases the applicability of RL to real-world problems.

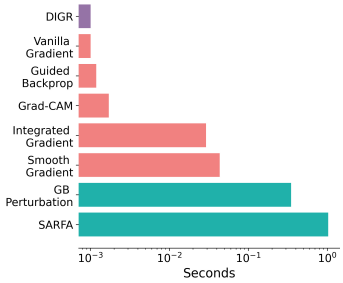
6.2 Background and Motivation

6.2.1 Policy Distillation

Policy distillation [Rusu et al., 2015, Czarnecki et al., 2019] transfers knowledge from one teacher policy π_t to a student policy π_s by training the student policy to produce the same



(a) Saliency Maps of Red-Fetch-Green



(b) Saliency Map Generation Time

Figure 6.1: (a). Different saliency maps on Red-Fecth-Green. All gradient-based saliency maps (Vanilla Gradient, Guided Backprop, Grad-CAM, Integrated Gradient and Smooth Gradient) produced by the PPO policy are noisy and show noticeable saliency on task-unrelated features. Gaussian-Blur Perturbation (GB Perturbation), SARFA saliency maps and saliency maps produced by DIGR approach demonstrate saliency on the red agent and green target object only. (b). The average time for each method to explain one action selection for states of Red-Fetch-Green during policy deployment with a CPU of Intel i7-9750H and a GPU of GeForce RTX 2080 Ti. We mark DIGR with purple and use red and green colors to represent normal gradient-based and perturbation-based saliency map methods.

behavior as the teacher policy. This is normally achieved by supervised regression to minimize the following objective:

$$J = \mathbb{E}_{s\sim\pi_c}[D(\pi_t(s), \pi_s(s))], \tag{6.1}$$

where π_c is the control policy that interacts with the environment to produce states for training, and D is a distance metric. There are multiple choices for both π_c and D. For example, the control policy π_c could take the form of the teacher policy π_t or student policy π_s or even a combination of them. Suitable distance metrics could be mean squared error or Kullback–Leibler divergence (KL divergence).

6.2.2 Saliency Map in RL

Addressing the interpretability of RL has attracted considerable attention in recent years. One common category of methods used visualization techniques such as saliency maps [Wang et al., 2016, Greydanus et al., 2018], attention mechanism [Mott et al., 2019] and object detection [Iyer et al., 2018] to explain deep neural network policies. Some other methods aimed to learn intrinsically interpretable policies in the formats of decision tree [Silva et al., 2020, Liu et al., 2021], programming language [Verma et al., 2018] or logic formulations [Zhang et al., 2021]. Besides the methods above, researchers also enhanced the understanding of RL decision making by using evidence-driven interpretation [Dao et al., 2018, 2021], contrastive explanations [Lin et al., 2020], counterfactual analysis [Rupprecht et al., 2019, Atrey et al., 2020] and state abstraction [Topin and Veloso, 2019]. In this work, we focus on saliency map explanations.

Saliency map techniques are popular in computer vision and RL communities for interpreting deep neural networks. Gradient-based methods calculate the gradient of some function f with respect to inputs s based on the chain rule and then use the gradients to estimate the influence of input features on the output. In RL, one common approach is the Jacobian saliency map [Wang et al., 2016] which computes the saliency of input feature s_i as $|\frac{\partial f(s)}{\partial s_i}|$ where function f could be calculated from either the state-action value $Q(s, a)$ in Q-learning or the action distribution $\pi(s)$ in actor-critic methods. Other gradient-based visualization methods from the field of image classification are also explored [Greydanus et al., 2018, Rosynski et al., 2020] but most of them didn't work well in the RL domain.

Perturbation-based methods compute the saliency of an input feature by perturbing (e.g. removing, altering or masking) the feature and observing the change in output. Given a state input s, a perturbed state s' could be generated by inducing a perturbation on input feature s_i. The approach of computing the change in output caused by the perturbation may vary

based on the form of RL agent. For example, in Q-learning, the network output is a scalar and thus the saliency of s_i could be defined as $|Q(s,a) - Q(s',a)|$. In actor-critic methods, the saliency of s_i could be defined as $D_{KL}(\pi(s)||\pi(s'))$ which is the KL divergence between action distributions before and after the perturbation. Alternatively, [Greydanus et al., 2018] considered the output of actor as a vector and computed the saliency as $\frac{1}{2}||\pi(s) - \pi(s')||^2$. Puri et al. [2020] further proposed an approach of Specific and Relevant Feature Attribution (SARFA) to address the specificity and relevance in perturbation-based saliency maps.

6.2.3 Motivation

We first introduce a simple fetching-object task in MiniGrid and demonstrate the results of different saliency map methods on this task to motivate our method. In the fetching-object task in MiniGrid, the environment is a room composed of 8x8 grids and 4 entities with unique colors. The red agent needs to locate and pick up the green object, while the yellow and blue objects are distractors. Based on the task rule, we name this task as Red-Fetch-Green. We first use PPO [Schulman et al., 2017] to train an RL policy to solve the task and then investigate the interpretability and computation efficiency of different saliency map methods to explain the policy. Examples of gradient-based (Vanilla Gradient [Simonyan et al., 2013], Guided Backprop [Springenberg et al., 2014], Grad-CAM [Selvaraju et al., 2017], Integrated Gradient [Sundararajan et al., 2017], Smooth Gradient [Smilkov et al., 2017]) and perturbation-based (Gaussian-Blur Perturbation [Greydanus et al., 2018] and SARFA [Puri et al., 2020]) saliency maps for Red-Fetch-Green are shown in Figure 6.1a. We also include an example of saliency maps generated by our DIGR approach for comparison. In general, perturbation-based saliency maps mainly demonstrate high saliency on task-relevant features (e.g. red agent and green target object) while gradient-based saliency maps are noisier and harder to interpret. However, the high quality of perturbation-based saliency maps is achieved with an increased cost of computation time. As shown in Figure 6.1b,

perturbation-based saliency map takes more time to generate compared to gradient-based saliency maps and counterfactual analysis of 'Represent And Mimic'(RAMi) [Liu et al., 2021]. The computation time of perturbation-based saliency maps is highly affected by the input size and policy network architectures. This makes it incompatible with many real-world tasks that require real-time interpretability such as autonomous driving. Thus, based on the result in Figure 6.1, we find that normal gradient-based saliency maps are computationally more efficient but hard to interpret while perturbation-based saliency maps are more interpretable but come with a higher computation cost during deployment. This finding motivates us to think about how we can keep the computation efficiency of gradient-based methods and the high interpretability of perturbation-based methods while avoiding their limitations, and thus propose DIGR.

How does DIGR generate interpretable saliency maps like perturbation-based methods while only requiring a short generation time as the most efficient Vanilla Gradient saliency maps? Is it possible for us to use gradient-based methods such as Vanilla Gradient method to generate high-quality saliency maps as those from perturbation-based methods? We answer these questions in the next section.

6.3 Method

Our approach to achieving both computational efficiency and high interpretability in RL is to produce a policy whose gradient-based saliency maps are comparable to those of perturbation-based methods. To achieve this, given a trained RL policy, we set its perturbation-based saliency maps as supervisory signals and update the weights of the policy so that its gradient-based saliency maps match the perturbation-based saliency maps. Since the computations involved in gradient-based saliency maps are differentiable, we can use stochastic gradient descent to conduct the training. The idea of optimizing gradient-based saliency

maps has a close connection with input gradient regularization which imposes constraints on how input gradients behave. For example, Ross and Doshi-Velez [2018] penalized input gradients based on an expert annotation to prevent the network from "attending" to certain parts of the input in an image classification task. Inspired by this, the training of the gradient-based saliency map in our approach is conducted by selectively penalizing the gradients of input features that have low perturbation-based saliency.

One challenge of selective input gradient regularization is that optimizing gradient-based saliency maps may also affect the policy output and thus degrade the task performance. To avoid this, we conduct policy distillation to ensure that the new policy maintains the same task performance. We give a more formal introduction of our method below.

Given an RL policy π and input s, we define the function g as the method used in generating gradient-based saliency map M_g and function f as the method used in generating perturbation-based saliency map M_p. Both M_g and M_p have the same size as input s. Each element in the saliency map, M_{g_i} and M_{p_i}, are computed as

$$\begin{aligned} g(s,i,\pi) &= \mid \sum_a \pi(a|s) \frac{\partial \pi(a|s)}{\partial s_i} \mid \\ M_{g_i} &= \frac{g(s,i,\pi)}{\max\limits_{0 \le j \le N} g(s,j,\pi)} \end{aligned} \tag{6.2}$$

$$\begin{aligned} f(s,i,\pi) &= D_{KL}(\pi(s) || \pi(m(s,i))) \\ M_{p_i} &= \frac{f(s,i,\pi)}{\max\limits_{0 \le j \le N} f(s,j,\pi)} \end{aligned} \tag{6.3}$$

where $g(s,i,\pi)$ and $f(s,i,\pi)$ compute the gradient-based and perturbation-based saliency values of input feature s_i given policy π. These saliency values are then normalized between

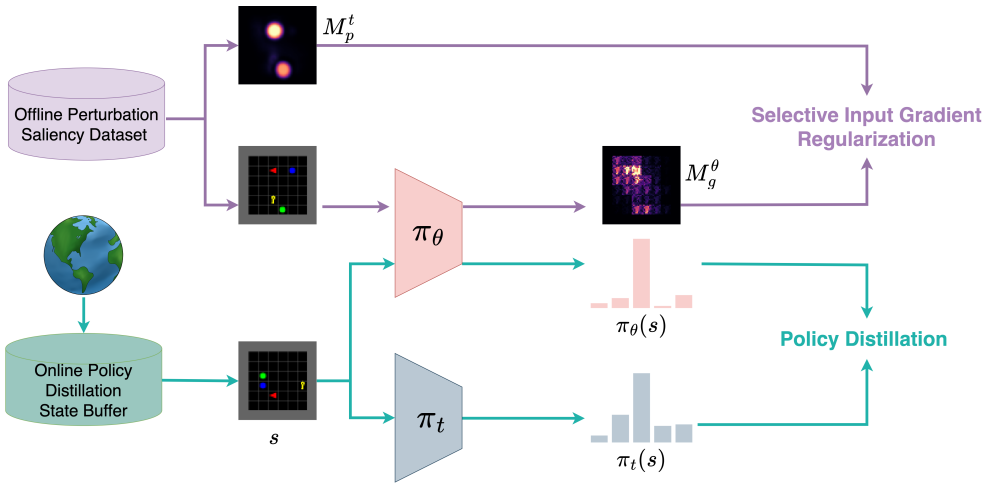


Figure 6.2: Framework of our approach. Policy π_θ is used as the control policy and interacts with the environment. The experienced states are saved into a replay buffer and then sampled later for policy distillation. The training includes two objectives. The first objective is using input gradient regularization to regularize gradient-based saliency map M_g^θ based on the perturbation-based saliency map M_p^t. The second objective is using policy distillation to make sure the learning policy π_θ has the same behavior as the trained policy π_t.

0 and 1 to form saliency maps that contain N elements in each map. In this work, perturbation function m induces a Gaussian blur on the input with the input feature of interest s_i as the center [Greydanus et al., 2018]. It's worth mentioning that, besides perturbation-based saliency maps, DIGR could be easily extended to utilize other saliency data (e.g. saliency maps from expert annotation) as supervisory signals. In this work, we focus on using perturbation-based saliency maps for input gradient regularization as they show high interpretability and can be computed as long as we have access to the policy and states.

After introducing the process of generating two types of saliency maps given an RL policy and state input, we introduce how they are used in DIGR. Given a trained RL policy π_t, DIGR aims to produce a new policy π_θ with parameters θ that can generate interpretable saliency maps using a gradient-based method. Given a state input s, the saliency map could differ based on the generation method (gradient-based vs perturbation-based) and the policy (π_t vs π_θ) used to generate them. For clarity, we define these 4 types of saliency maps as ${M_g}^t$, ${M_g}^\theta$, ${M_p}^t$, ${M_p}^\theta$ where the subscript of g represents gradient-based saliency maps and p represents perturbation-based saliency maps. The superscript of t represents

the saliency map is generated by the original teacher policy π_t and θ represents the saliency map is generated by policy trained with DIGR (π_θ). In DIGR, we use the perturbation-based saliency maps generated by the teacher policy (${M_p}^t$) to provide supervisory signals to regularize the gradient-based saliency maps generated by the DIGR policy (${M_g}^\theta$). Then the loss function for input gradient regularization is

$$L = \mathbb{E}_{s\sim d_{\pi_\theta}}[\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{[0,\infty)}(\lambda - {M_p}_i^t) \times {M_g}_i^\theta] \tag{6.4}$$

where d_{π_θ} is the state distribution following policy π_θ and N is the number of input features in the saliency map. ${M_p}^t$ and ${M_g}^\theta$ have the same size and are both indexed by i. Threshold λ is used in the indicator function $\mathbb{1}$ to determine whether one input gradient should be penalized. The indicator function $\mathbb{1}$ returns 1 if $\lambda - {M_p}_i^t \geq 0$ and 0 otherwise. In other words, if the perturbation-based saliency for an input feature is below threshold λ, the loss penalizes its gradient-based saliency. This selective penalization allows the model to only keep high saliency on task-relevant features selected by the perturbation-based saliency maps.

The final loss function in our approach is a weighted combination of selective input gradient regularization and policy distillation. In practice, generating perturbation-based saliency maps online for input gradient regularization could be time-consuming and slow down the overall training. To address this, we build an offline perturbation saliency dataset D which contains states sampled from d_{π_t} and the corresponding perturbation-based saliency maps generated in advance. Because of the policy similarity brought by policy distillation, we use D to approximate d_{π_θ} for input gradient regularization. As a result, the loss function for

DIGR is

$$L_{DIGR} = \mathbb{E}_{s\sim D}[\underbrace{\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{[0,\infty)}(\lambda - M_{p_i}^{t}) \times M_{g_i}^{\theta}]}_{\text{Input Gradient Regularization}} + \alpha \underbrace{\mathbb{E}_{s\sim d_{\pi_\theta}}[D_{KL}(\pi_t(s)||\pi_\theta(s))]}_{\text{Policy Distillation}} \tag{6.5}$$

where α is a weighting parameter used to balance the loss of input gradient regularization and policy distillation. We show the complete architecture of our approach in Figure 6.2.

6.4 Experimental Results

We conduct experiments on three tasks including Red-Fetch-Green in MiniGrid, Breakout in Atari games and CARLA Autonomous Driving to demonstrate the effectiveness of our approach. In Red-Fetch-Green, the red agent needs to locate and pick up the green object while avoiding picking up other distractors in a room composed of 8x8 grids. In Breakout, the paddle is controlled to move at the bottom to ricochet the ball against the bricks and eliminate them for rewards. Besides these two tasks, we designed a CARLA Autonomous Driving task in which the agent needs to control an autonomous car driving on a highway while avoiding collisions. Since CARLA's simulation clock can be matched with the real time, we use it to show how the high quality and computation efficiency of our approach in interpreting RL policies could be important in real-world scenarios.

6.4.1 Setup

RL Training

In our experiments, we first use PPO algorithm to train RL policies on Red-Fetch-Green, Breakout and CARLA Autonomous Driving. The trained RL policies, which are used to generate offline perturbation saliency datasets for input gradient regularization, also serve as the teacher policy in policy distillation and generate saliency maps for comparison. In all three tasks, we used similar network architectures composed of 3 convolutional layers and 2 linear layers but with different layer sizes. The trained RL policies achieved reasonably good performance in each task: The policy in Red-Fetch-Green solves the task with a success rate of 100%; the policy in Breakout achieves an average score of 320; the policy in CARLA Autonomous Driving could drive smoothly and learned to steer to avoid collision with other vehicles. We include more details of RL training in the appendix.

Offline Perturbation Saliency Dataset

To conduct selective input gradient regularization, we generate an offline perturbation saliency dataset by sampling states experienced by the trained RL policy π_t and generating the corresponding Gaussian-Blur perturbation saliency maps [Greydanus et al., 2018]. The perturbation saliency datasets of Red-Fetch-Green, Breakout, and CARLA Autonomous Drivinghe 1k, 10k, and 2.5k pairs of states and saliency maps. Although our method still needs to generate perturbation-based saliency maps, the computation happens in the training stage without affecting the computation efficiency during deployment. Also, the computation problem could be mitigated by the limited size of the dataset (e.g. 1k, 10k, and 2.5k states in Red-Fetch-Green, Breakout, and CARLA respectively) and the potential utilization of parallel computing with multiple machines.

DIGR Training

DIGR uses selective input gradient regularization and policy distillation to produce a new policy that achieves efficient interpretability while maintaining task performance. In all three experiments, we randomly initiate the new policy π_θ. To further stabilize the training, we consider the training of selective input gradient regularization and policy distillation as a multi-objective optimization problem and used the technique of projecting conflicting gradients (PCGrad) [Yu et al., 2020] to mitigate gradient interference. More hyperparameters of training are included in the appendix.

6.4.2 Effectiveness via Visual Illustrative Examples

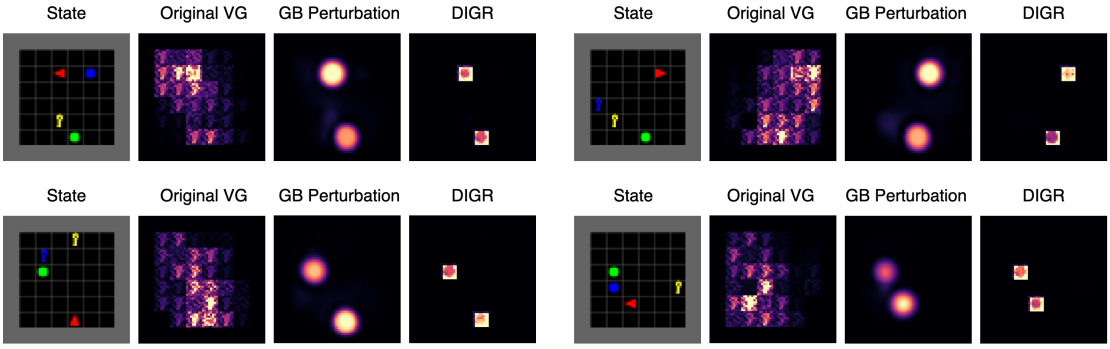


Figure 6.3: Demonstration of our approach on Red-Fetch-Green. There are four sets of examples and each set includes a state, a Vanilla Gradient saliency map generated by the original policy (Original VG), a Gaussian-Blur perturbation-based saliency map (GB Perturbation) generated by the original policy and a Vanilla Gradient saliency map generated by the policy trained with DIGR. The annotation of DIGR on the figure refers to Vanilla Gradient saliency maps generated by the policy trained with DIGR. In all examples, GB Perturbation and DIGR saliency maps show high saliency on the red agent and green target while Original VG saliency maps are noisy and hard to interpret.

The main goal of our approach is to allow RL policies to generate interpretable saliency maps with computationally efficient gradient-based methods. To demonstrate the effectiveness of our approach, we provide examples of the most computationally-efficient Vanilla Gradient saliency maps before and after our method, and Gaussian-Blur perturbation saliency maps

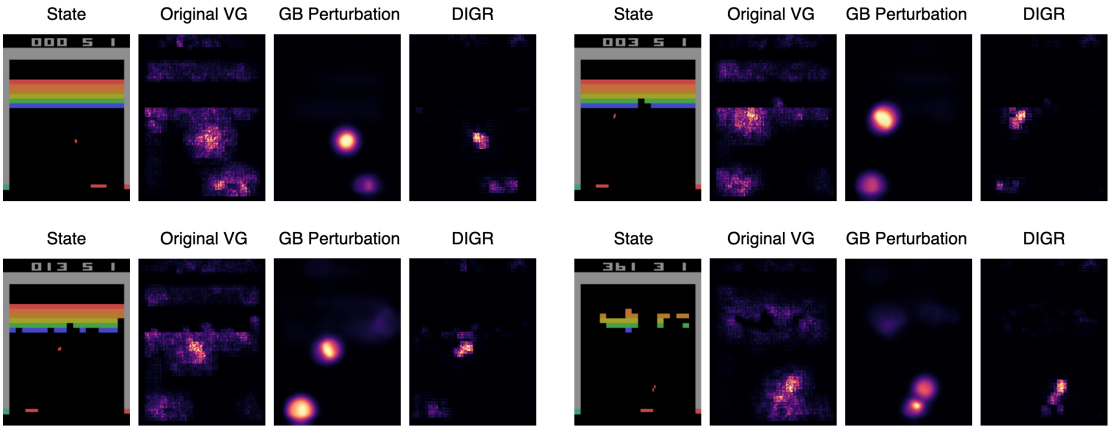


Figure 6.4: Demonstration of our approach on Breakout. VG and GB Perturbation stand for Vanilla Gradient and Gaussian-Blur Perturbation. Both DIGR and Gaussian-Blur perturbation-based saliency maps demonstrate high saliency mainly on the paddle and ball while the Vanilla Gradient saliency maps generated by the original policy (Original VG) are noisier.

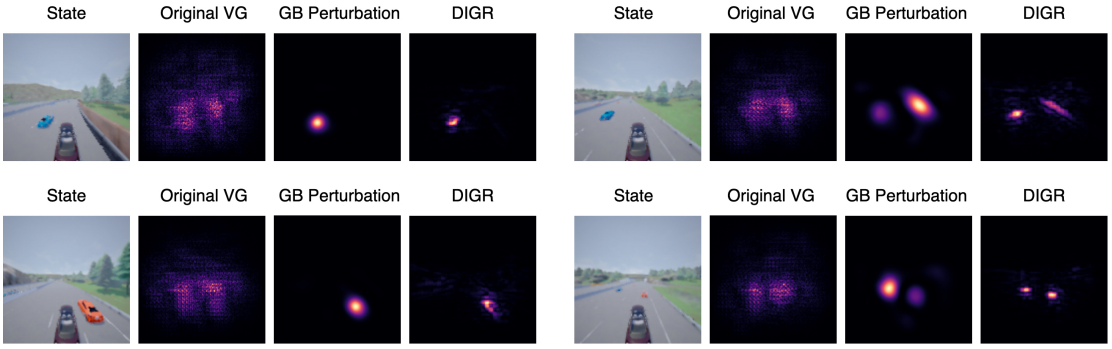


Figure 6.5: Demonstration of our approach on CARLA Autonomous Driving. VG and GB Perturbation stand for Vanilla Gradient and Gaussian-Blur Perturbation. In the left two sets of examples, DIGR and GB Perturbation methods demonstrate high saliency on the vehicles that got close to the controlled vehicle. In the top-right example, DIGR and GB perturbation methods show high saliency on the vehicle and road curb. In the bottom-right example, DIGR and GB perturbation methods show high saliency on two vehicles ahead. DIGR and GB perturbation methods didn't show saliency on the controlled vehicle because the controlled vehicle is always at the same region of the images for all states and is not salient to the performance. The saliency is demonstrated on other features that may lead to a collision and affect the performance. In all four sets of examples, Vanilla Gradient saliency maps generated by the original policy (Original VG) are very similar and hard to distinguish.

that work as supervisory guidance in Figures 6.3, 6.4, and 6.5.

Our results show that Vanilla Gradient saliency maps generated by original RL policies are noisy and hard to interpret. However, after the optimization with our approach, we can use

the same saliency map method to generate much more interpretable saliency maps which reduces a large amount of unexplainable saliency and demonstrate high saliency on task-relevant features only. The saliency maps generated by our approach also have a close similarity to Gaussian-Blur perturbation-based saliency maps which demonstrates the successful saliency guidance. We provide more visual examples containing saliency maps produced by other gradient-based methods for comparison in the appendix.

6.4.3 Importance of Computational Efficiency

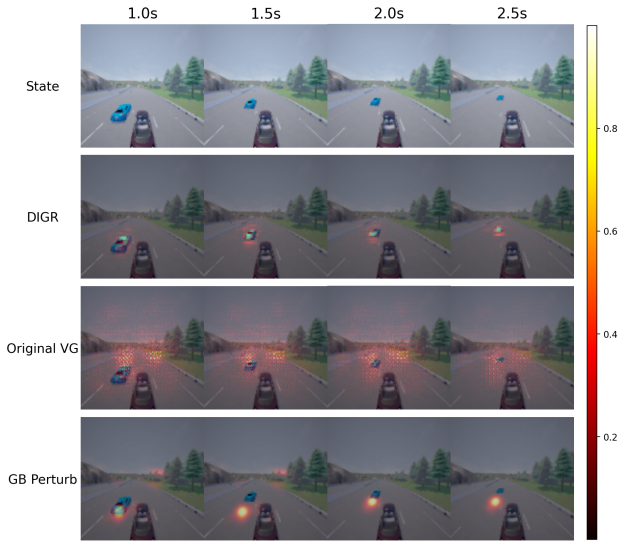


Figure 6.6: Different types of saliency maps on a sequence of states in CARLA Driving. Vanilla Gradient saliency maps generated by the policy trained with DIGR always demonstrate high saliency on the traffic vehicles while Vanilla Gradient saliency maps generated by the original policy (original VG) are noisy and just show saliency in the center region of all states. Gaussian-Blur perturbation-based saliency maps show saliency behind the vehicle because of the computation delay. The bar on the right represents the mapping between saliency values and colors.

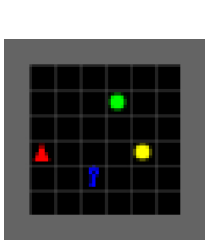
In this section, we further show the importance of our approach by demonstrating that missing either computation efficiency or high interpretability makes it difficult to achieve interpretable RL in real-world scenarios. We take Autonomous Driving as an example and show the results of utilizing different saliency maps to explain a sequence of RL decision

making in Figure 6.6. In our experiments, the state of CARLA Autonomous Driving is a 128x128 RGB image taken every 0.05 seconds by a camera attached to the ego vehicle. Although Gaussian-Blur perturbation-based saliency maps show high interpretability as seen in Figure 6.5, it takes 0.97±0.02 seconds to generate one saliency map with a GPU of RTX 2080Ti. This means there's a delay of almost one second between meeting the state and the availability of the corresponding saliency map and all saliency maps for states experienced during the delay will be missed. In contrast to Gaussian-Blur perturbation-based saliency maps each takes 0.97 seconds to generate on average, Vanilla Gradient saliency maps are much more efficient to compute and take only 0.0021±0.0001 seconds for each state with the same machine. However, Vanilla Gradient saliency maps generated by normal RL policies are hard to interpret and only our approach achieves both computation efficiency and high interpretability.

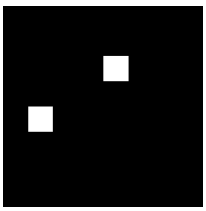
6.4.4 Saliency Dataset and Evaluation

Besides illustrative examples, we also aim to provide a quantitative evaluation of saliency maps generated by different approaches and thus introduce a new saliency dataset based on Red-Fetch-Green. Different from previous work that relies on expert annotations and classifies each state element as either an important or unimportant feature [Puri et al., 2020], we focus on features whose saliency importance is certain. There are six types of objects in Red-Fetch-Green including the red agent, the green target object, the blue and yellow distractors, grey walls, and black empty grids. Based on the roles of objects, we assume the red agent and green target are important features as they have the most important information required for optimal decision making and assume the empty tiles as unimportant features since they do not provide any information. The two distractors and grey walls are not included in the dataset because their influence on decision making is either uncertain or only exists in a small subset of state space. We collected 10k states in the saliency dataset

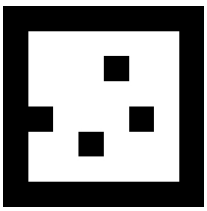
and provide an example in Figure 6.7.



(a) state



(b) important saliency



(c) unimportant saliency

Figure 6.7: **a**. An example state in the saliency dataset of Red-Fetch-Green. **b**. Regions whose saliency is important. **c**. Regions whose saliency is unimportant.

	Saliency on Red-Fetch-Green		
	important	unimportant	AUC
VG	56.04	278.10	0.840
Guided BP	**82.84**	35.67	0.993
Grad-CAM	43.12	364.97	0.686
Smooth G	**83.05**	84.76	0.991
Integrated G	67.79	232.09	0.900
GB Perturbation	**86.11**	77.81	0.989
SARFA	58.40	42.17	0.895
DIGR	**72.52**	**0.00**	**0.997**

Table 6.1: Saliency results of Vanilla Gradient (VG), Guided Backpropagation (Guided BP), Grad-CAM, Smooth Gradient (Smooth G), Integrated Gradient (Integrated G), Gaussian-Blur Perturbation (GB Perturbation), SARFA of the original policy and Vanilla Gradient of DIGR policy on Red-Fetch-Green. Our method keeps a comparable amount of important saliency, reduces all unimportant saliency, and achieves the highest AUC.

To evaluate the quality of different saliency maps, we compute the average amount of important saliency and unimportant saliency in each saliency map. Furthermore, we also compare different saliency maps with Area under the Receiver Operating Characteristic Curve (AUC), which is a popular metric used to evaluate saliency maps [Iyer et al., 2018, Puri et al., 2020]. As shown in Table 6.1, our approach keeps a comparable amount of important saliency, reduces all unimportant saliency and achieves the highest AUC compared with other approaches. The decreased amount of unimportant saliency is in line with our expectation since our approach works by penalizing the saliency that is not helpful for interpretation.

As a result, our approach utilizes gradient-based and perturbation-based saliency maps for training and finally achieves even better saliency maps.

6.4.5 Policy Performance Maintenance

The objective of optimizing gradient-based saliency maps may change the action selection of the original policy and thus cause the policy performance to degrade. In DIGR, we use policy distillation to constrain the output of the new RL policy to remain close to the original policy. To verify its effectiveness, we plot the performance of DIGR policy during training and compare it with the results of the original policy. As seen in Figure 6.8, the policy trained with our approach could achieve similar performance as the original policy.

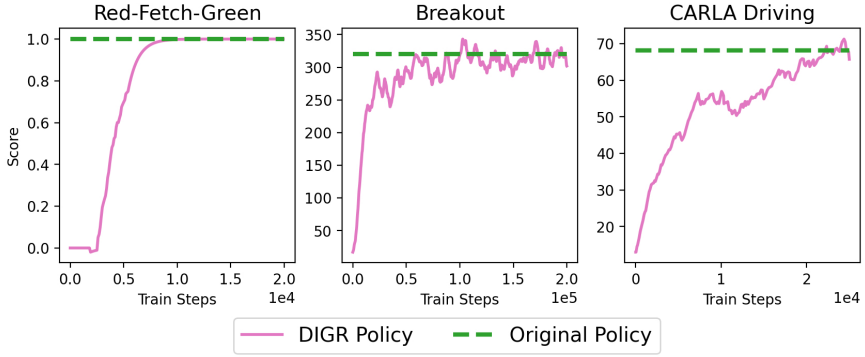


Figure 6.8: The performance of DIGR policy could match the performance of the original policy.

6.4.6 Improved Robustness to Attacks

Due to the importance of robustness of neural networks [Zheng et al., 2016, Cheney et al., 2017, Carlini and Wagner, 2017] and recent research findings of a deep entanglement between adversarial attacks and interpretability of deep neural network (DNN) models [Tao et al., 2018, Ignatiev et al., 2019], we are also interested in DIGR's influence on policy's robustness to attacks. To study that, we evaluate the robustness of RL policies before and after applying

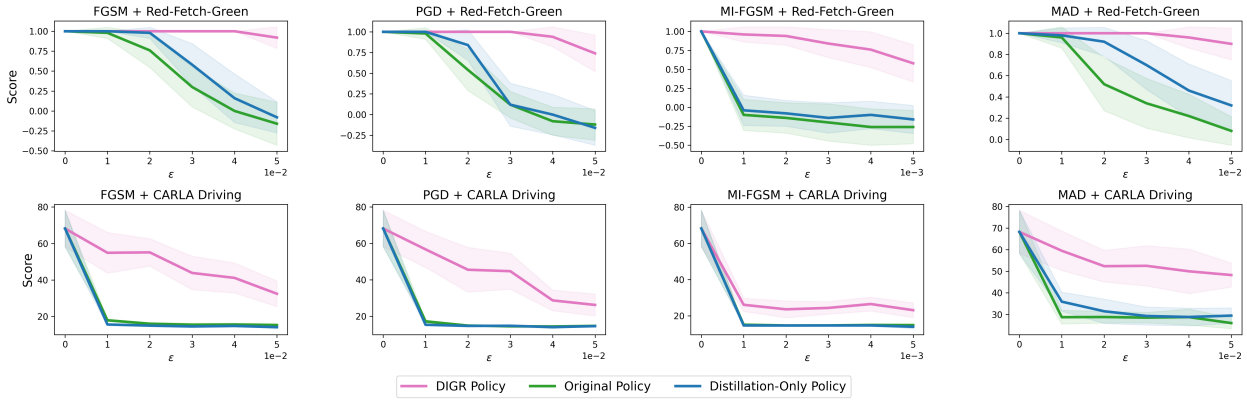


Figure 6.9: Policies trained with DIGR achieve much stronger robustness to all four types of adversarial attacks (FGSM, PGD, MI-FGSM and MAD) compared to the policies trained with normal RL algorithms. Although policy distillation also helps robustness slightly, selective input gradient regularization makes the most contribution to the improved robustness. All results are averaged over 50 runs in Red-Fetch-Green and 20 runs in CARLA Autonomous Driving. The shaded area represents one standard deviation.

DIGR to four types of adversarial attacks including Fast Gradient Sign Method (FGSM) [Huang et al., 2017], Projected Gradient Descent (PGD) [Madry et al., 2018], Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [Dong et al., 2018] and Maximum Action Difference (MAD) [Zhang et al., 2020] in Red-Fetch-Green and CARLA Autonomous Driving tasks. Since both policy distillation and input gradient regularization in our approach could affect the robustness of RL policies, we further include an ablation study by conducting policy distillation only to understand their own influence on robustness. As shown in Figure 6.9, our approach significantly improves the robustness of RL policies. Although policy distillation also improves the robustness slightly, selective input gradient regularization contributes the most to the significant robustness gains.

6.5 Conclusion

We propose an approach called DIGR to improve the efficient interpretability of RL by re-training a policy with selective input gradient regularization and policy distillation. Our

approach allows RL policies to generate highly interpretable saliency maps with computationally efficient gradient-based methods. We further show that our approach is able to improve the robustness of RL policies to multiple adversarial attacks. Interpretable decision-making and robustness to attacks are two challenges in deploying RL to real-world systems. We believe our approach could help to build trustworthy agents and benefit the deployment of RL policies in practice.

Chapter 7

Linking Global Top-Down Views to First-Person Views in the Brain

(This chapter is reprinted with permission, from Jinwei Xing, Elizabeth R Chrastil, Douglas A Nitz and Jeffrey L. Krichmar. Linking global top-down views to first-person views in the brain. Proceedings of the National Academy of Sciences 119.45 (2022): e2202024119. @PNAS

7.1 Introduction

Humans are able to translate their location and navigational goals on an external map into decision making behaviors in the environment. A glance at a map can help place you in your local surroundings. Conversely, when looking at one's local surroundings, one can place oneself on a global map. The ability to seamlessly move between top-down (TDVs) and first-person views (FPVs) may be important for navigation and memory formation, as well as many cognitive tasks (e.g., building a cabinet from a plan drawn on paper, or finding an

extra screw after the cabinet is constructed and referring back to the plan to find out where the screw should go). Evidence from other animals suggests that they also have the ability to translate their position from one spatial reference frame to another [Alexander and Nitz, 2015, Meister and Buffalo, 2018, Meister, 2018, Geva-Sagiv et al., 2015]. In particular, bats appear to have the ability to take translate a TDV while flying above the landscape to a FPV when navigating on the ground or foraging for food [Ulanovsky and Moss, 2007, Omer et al., 2018, Sarel et al., 2017, Geva-Sagiv et al., 2015].

Studies suggest that the entorhinal cortex (EC), retrosplenial cortex (RSC), subiculum (SUB), posterior parietal cortex (PPC), and hippocampus (HPC) could play significant roles in linking locations and orientations relative to one view to locations and orientations relative to another [Alexander and Nitz, 2015, Byrne et al., 2007, Chrastil et al., 2018, Clark et al., 2018, Epstein et al., 2017, Oess et al., 2017, Meister and Buffalo, 2018]. The computations and neural implementations that manifest this cognitive ability have scarcely been addressed despite numerous navigation experiments in humans and rodents. Computational modeling suggests that these transformations and linkages could be accomplished through specific encoding of parameters [Bicanski and Burgess, 2018, Byrne et al., 2007, Oess et al., 2017], or mixed selectivity that responds to multiple variables [Rounds et al., 2018]. However, it is unclear whether mechanisms for linkage and transformation among perspectives operate to form a single mapping of location from both perspectives, or serve to link analogous locations in two different mappings. Mapping of location and orientation is robustly observed in the rodent EC, HPC and SUB [Danjo et al., 2018, Derdikman and Moser, 2010, Kim et al., 2012, Sharp, 1997], which provide input to RSC and other brain regions. The PPC provides egocentric information to the RSC [Wilber et al., 2014]. Furthermore, the visual system plays an important role in driving spatial activity [Meister, 2018]. Still, the exact role of these brain regions and their neural computations, especially in the context of viewpoint transformations, remain poorly understood.

In the present article we attempt to answer the following open questions: 1) What architectures might support these transformations and linkages? 2) What are the computations and neural implementations underlying linkages and transformations between TDVs and FPVs? 3) What cues or landmarks are required to make these transformations and linkages? To answer these questions, we take a model-free approach by using Variational AutoEncoders (VAEs) to reconstruct the FPV from the TDV of a robot simulation, and vice versa [Kingma and Welling, 2019]. The latent variables, which make a transformation between the encoding network layers and the decoding network layers, will be compared with brain responses.

In contrast to neurobiologically inspired models of these transformations [Bicanski and Burgess, 2018, Byrne et al., 2007], which suggest that the computations in each direction utilize the same circuit by inverting the transformation operation, our results indicate that each direction of transformation is dictated by different computations carried out by separate circuits. Whereas going from a TDV to FPV required specific representations of place and objects, going from a FPV to a TDV tended to use mixed representations and strong head direction signaling. In both cases, one view was accurately reconstructed from the other. In addition, both neural codes were flexible and adaptive to perturbations. We suggest that this is a possible neural implementation that could support important navigation functions.

7.2 Results

7.2.1 Robot Simulation and Modeling Transformations

To test the ability to link TDV to FPV and vice versa, data was collected with the Webots [Webots, Michel, 2004] robot simulation environment (Fig. 7.1A). The simulated robot was a Khepera with a camera, and proximity sensors to detect objects and boundaries. The robot freely explored its space. Approximately every second, the overhead view of simulation

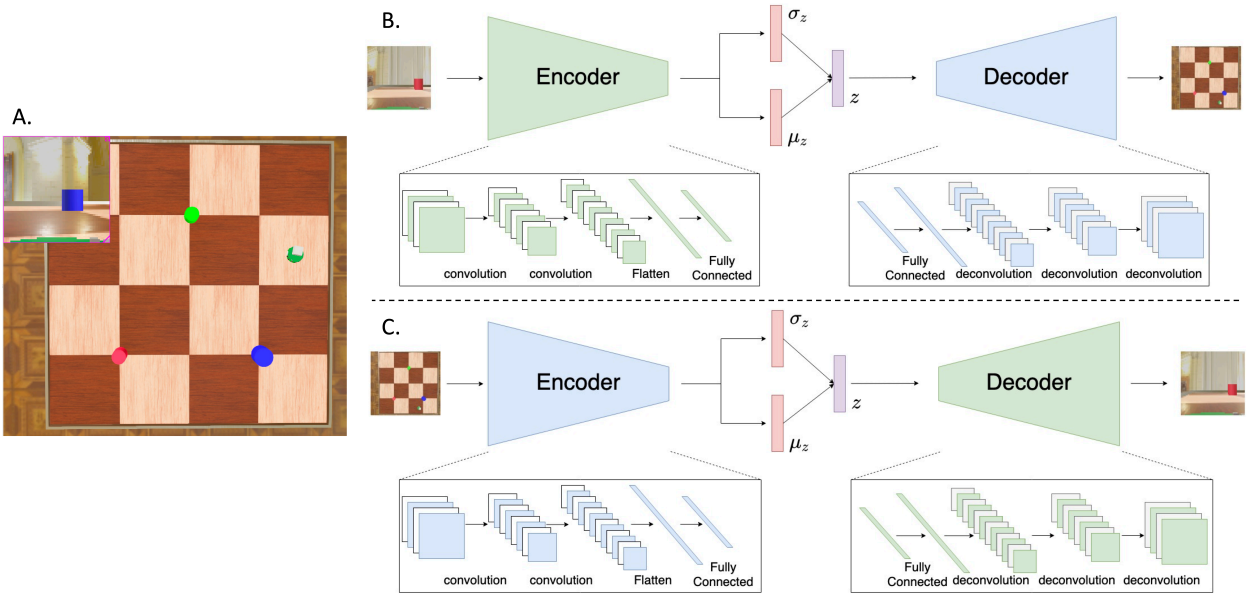


Figure 7.1: Simulation setup and model architectures. A. Robot freely explored a square arena, which had 3 colored cylinders. The robot is located on the middle right facing the blue cylinder. The inset shows the robot's camera view. *Note the camera view did not overlay the top-down images during data collection.* Robot was simulated using Webots [Webots, Michel, 2004]. B and C. Variational AutoEncoders (VAE) reconstruct images from robot simulation. The latent variables between the Encoder and Decoder are analyzed to understand the transformations and linkages between views. B. Takes first-person view as input and reconstructs top-down view. C. Takes top-down view as input and reconstructs first-person view.

(TDV), the robot's camera image (FPV), position, heading, and distance to the 3 cylinders were saved. 10,000 data points were collected: 8000 for training and 2000 for testing. In all conditions, even when environmental conditions changed (e.g., removing an object or changing the background) the robot position, heading and trajectory were identical for the 10,000 data points.

Two VAEs were constructed; one for reconstructing the TDV of the simulation environment from the FPV of the robot (Fig. 7.1B), and another for reconstructing the FPV from the TDV (Fig. 7.1C). The number of latent variables (μ, σ, and z in Figures 1B and 1C) varied from 30, 50, and 100. In the results presented below, only the 2000 testing data points were used for analysis.

The VAE was able to reconstruct a top-down view (TDV) from a first-person view (FPV) and vice versa. Figure 7.2 shows how the reconstructions improved as the loss decreased during training with 100 latent variables. After 20,000 epochs of training, the median reconstruction losses for the FPV to TDV and the TDV to FPV transformations were less than 0.01. The process was repeated five times with different random number generator seeds. All 5 runs for each number of latent variables were used for analysis. Supplemental figures 1-3 show the loss for simulations with 30, 50, and 100 latent variables. Although the medians were roughly similar for both types of transformations (e.g., with 100 latent variables, the median was 0.0078 for FPV to TDV and 0.0084 for TDV to FPV in supplemental figure 3), the TDV to FPV transformation had more outliers (i.e., images it had difficulty reconstructing). Because of this, the two distributions for all numbers of latent variables were significantly different ($p < 0.000001$; Wilcoxon sign rank test). Supplemental figures 4 and 5 show examples of the reconstructions after training.

7.2.2 Spatial Representations in Latent Variables

We wondered whether the latent variables of the VAEs had similar qualities to spatial representations observed in RSC, HPC, and SUB recordings. Indeed, many latent variables were sensitive to place, heading and objects. We looked at how well a latent variable correlated with the robot's distance to a cylinder, to an idealized head direction cell (cosine tuning curve with one of 16 preferred directions), or to an idealized place cell (2D Gaussian with one of 16 preferred locations). Table 7.1 shows the percentage of significant correlations in each case. The TDV to FPV transformations had significantly more latent variables that were strongly correlated with distance to the cylinder objects, and significantly more latent variables strongly correlated with place fields than the FPV to TDV transformations. In contrast, the FPV to TDV transformations had significantly more latent variables sensitive to head direction than the TDV to FPV transformation. Supplemental Figure 6, which

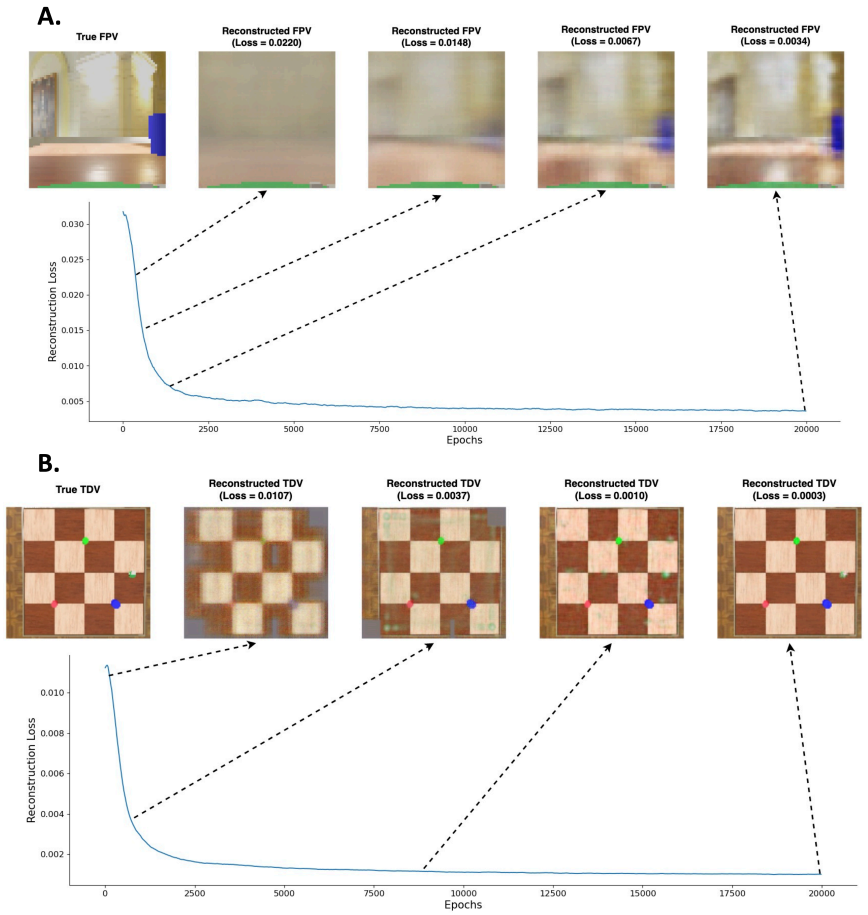


Figure 7.2: Reconstruction loss during VAE training with 100 latent variables. The true image is the VAE target and the other images are reconstructions at different points in the training. A. TDV to FPV transformation. B. FPV to TDV transformation.

Table 7.1: Percentage of strong correlations ($p < 0.01$). Asterisk denotes significantly more strong correlations for that transformation direction ($p < 0.01$; Wilcoxon rank sum test).

LV	Hdg (FPV to TDV)	Hdg (TDV to FPV)	Obj (FPV to TDV)	Obj (TDV to FPV)	Plc (FPV to TDV)	Plc (TDV to FPV)
30	49%*	33%	50%	73%*	50%	73%
50	54%*	26%	49%	73%*	47%	70%*
100	55%*	30%	49%	67%*	47%	69%*

plots the correlations for all latent variables, shows these trends, especially in the tails of the distributions where latent variables were strongly positively and negatively correlated.

Fig. 7.3 shows representative latent variable examples of head direction and location-specific tuning. Interestingly, the place fields for TDV to FPV tended to be sharp, whereas the FPV to TDV tended to be more diffuse. Even though there were fewer head direction sensitive latent variables in the TDV to FPV transformation, the shape of the head direction latent variables was similar for both transformation directions (see Supplemental Figures 7 and 8 for more examples). In contrast, FPV to TDV latent variables that were strongly correlated with place tended to be more diffuse than those in the TDV to FPV direction. For example, compare the FPV to TDV place fields in Supplemental Figure 11 to the TDV to FPV place fields in Supplemental Figure 12.

To understand the difference in spatial coding between transformation directions, we measured the spatial information [Skaggs et al., 1996] and spatial coherence [Kubie et al., 1990] of the latent variables. In general, spatial information measures the extent to which activity rates are high across a small subset of locations and low or non-existent across the remainder of an environment. Coherence measures the extent to which high activity rates cluster in a single location, as in "place fields" of HPC neurons. Together they provide a good metric for how strongly the latent variables encode locations. Figure 7.4 shows the distributions for these spatial metrics in simulations with 100 latent variables. For both transformation directions, the spatial information and spatial coherence were significantly larger than a random distribution containing the same latent variables with their positions shuffled ($p <$

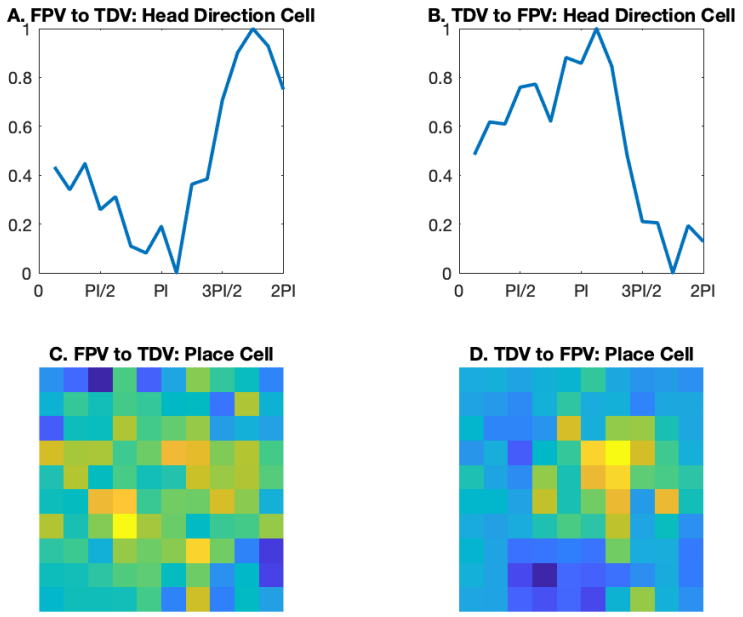


Figure 7.3: Representative latent variable responses during simulations with 100 latent variables. A and B. Latent variables that responded similarly to head direction cells. C and D. Latent variables that responded similarly to place cells. Note that C was typical of a First-Person to Top-Down view transformation and D was typical of a Top-Down to First-Person view transformation.

0.000001; Wilcoxon sign rank test). Furthermore, the spatial metrics were significantly larger for the TDV to FPV transformation than the FPV to TDV transformation ($p < 0.000001$; Wilcoxon sign rank test). Latent variable place fields were distributed throughout the environment with some tendency for the centers of place fields to be on the borders and corners (Supplemental Figure 36). The sparsity metric [Skaggs et al., 1996], which is roughly the fraction of the environment that the latent variable was active, ranged from very specific to broad (Supplemental Figure 37). Together, these measures suggest that both VAEs had strong spatial tuning and that the TDV to FPV had stronger spatial tuning than FPV to TDV. Overall, these results suggest that the TPV to FPV transformation relied more on place specific coding, whereas the FPV to TPV transformation relied more on head direction coding with diffuse place fields.

Spatial representations, such as place cells, grid cells, and head direction cells appear early

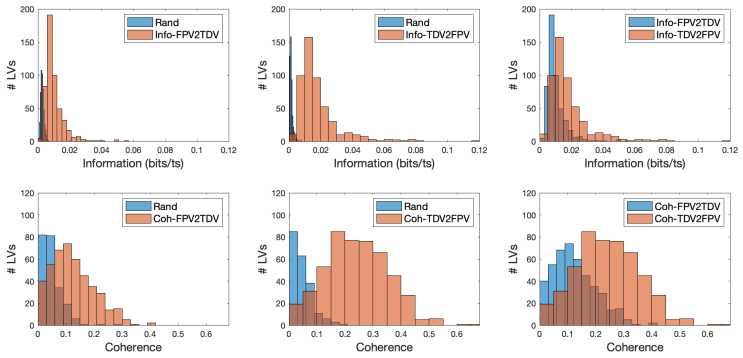


Figure 7.4: Spatial metrics for latent variables. The top row shows the distributions of spatial information and the bottom row shows the distributions of spatial coherence for simulations with 100 latent variables. The left column compares the spatial metrics for FPV to TDV transformations (orange) with a random distribution (blue) in which the location activity bins were shuffled. The middle column compares the spatial metrics for TDV to FPV transformations (orange) with a random distribution (blue). The right column compares TDV to FPV transformations (orange) with FPV to TDV transformations (blue). The TDV to FPV transformation had significantly stronger spatial metrics then the FPV to TDV transformation for both information and coherence.

in rodent development [Wills et al., 2010] and almost immediately upon entering a new environment [Frank et al., 2004]. We looked at the spatial metrics of the latent variables in the VAE model at 20, 200 and 2,000 epochs of training (Fig. 7.5). At each of these training stages, the spatial information and coherence were significantly larger than random in both transformation directions (Wilcoxon signed rank test $p < 0.001$). At 20 epochs, the spatial information was bimodal with many latent variables close to zero. By 200 epochs and continuing to the end of training (20,000 epochs), the spatial information had a high degree of overlap with the end of training (Fig. 7.5 top). After 20 epochs of training coherence was extremely low. Similar to spatial information, from 200 epochs until the end of training, the coherence had a high degree of overlap with the end of training (Fig. 7.5 bottom). Supplemental figures 43 through 54 show example latent variables that respond to place and head direction after 20, 200, and 2,000 training epochs. Given the spatial metric values and low reconstruction loss by 200 training epochs, it appears the model can support spatial navigation after limited environmental exposure.

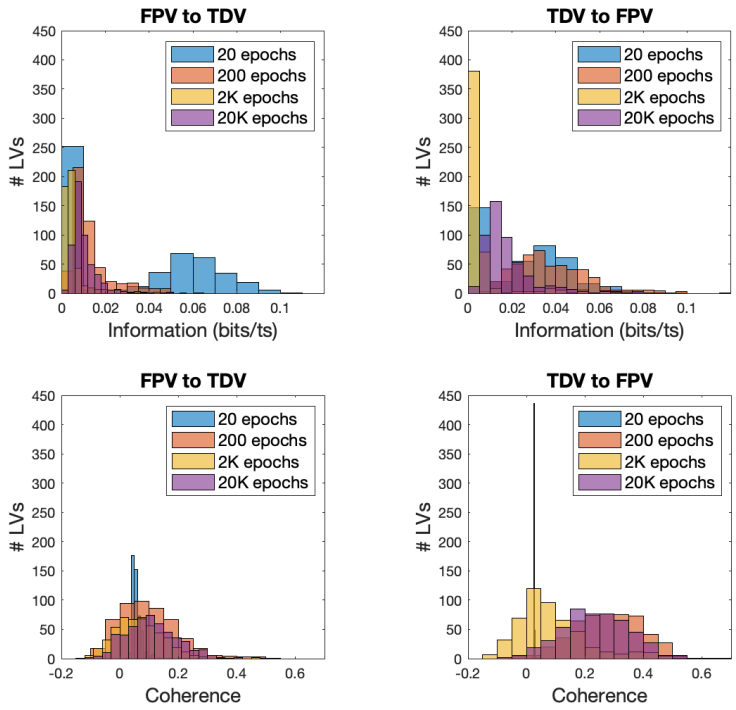


Figure 7.5: Spatial metrics for latent variables during early training. The top row shows the distributions of spatial information and the bottom row shows the distributions of spatial coherence for simulations with 100 latent variables after 20, 200, 2,000, and 20,000 epochs of training.

7.2.3 Effect of Latent Variable Ablations

We next tested how sensitive reconstruction was to particular latent variables. We ablated (i.e., set to zero) the most sensitive (top 25%) latent variables to environmental features. Fig. 7.6A shows the relative FPV to TDV reconstruction losses and Fig. 7.6B shows the relative TDV to FPV reconstruction losses. Relative loss was calculated by dividing the ablation loss by intact loss for each image. For the sensitivity to cylinders, head direction and location, the loss was significantly greater for the TDV to FPV transformation when the top 25% latent variables were ablated. This may be due to different representation schemes; whereas TDV to FPV rely more on specific selectivity, which could be sensitive to ablations, FPV to TDV may rely on a more distributed population code.

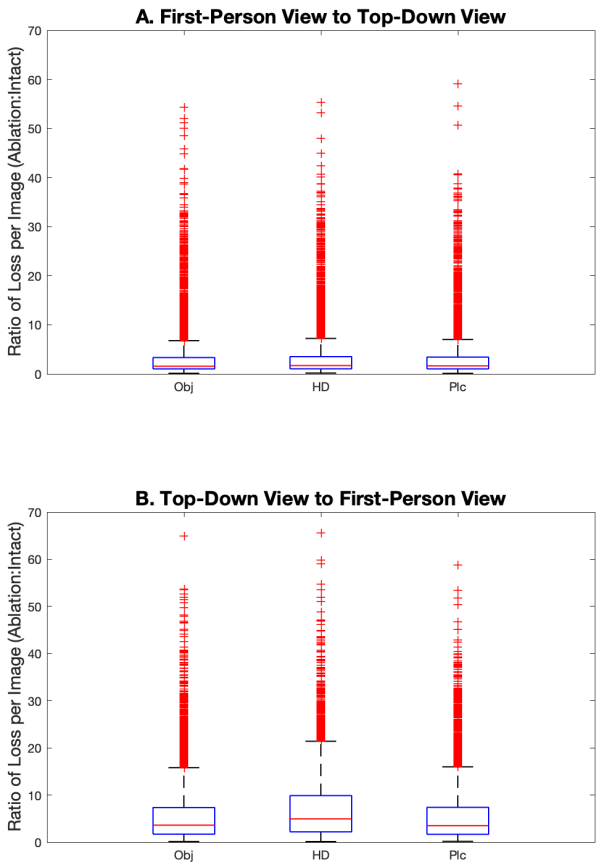


Figure 7.6: Relative loss during ablation studies of the top 25% latent variables that were correlated with objects (Obj), heading (HD), or place (Plc). The figures show the ratio of the ablation loss to the intact model loss for each image. In each box, the central mark is the median (red), the edges of the box are the 25th and 75th percentiles (blue), the whiskers extend to the most extreme data points that are not considered outliers, and the outliers are plotted individually with a red plus sign.A. FPV to TDV transformation. B. TDV to FPV transformation. All reconstruction losses for ablations were significantly larger for TDV to FPV than for FPV to TDV transforms ($p < 0.0000001$, Wilcoxon sign rank test).

7.2.4 Effect of Environmental Perturbations

We wondered how the VAEs would respond to perturbation of local and distal cues in the environment. Therefore, we ran simulations where the robot took the same trajectory for 10000 data points, but some aspect of the environment was changed. For example, the original background, which was a ballroom (Fig. 7.1A), was changed to mountains while

leaving everything else the same. This was an example of perturbing distal cues. In the other cases, we perturbed local cues by rendering the green cylinder to be invisible, or by rendering both the green and blue cylinders invisible. We then examined how the VAE, which was trained on the original environment, responded to the 2000 test data points in the perturbed environment (Fig. 7.7).

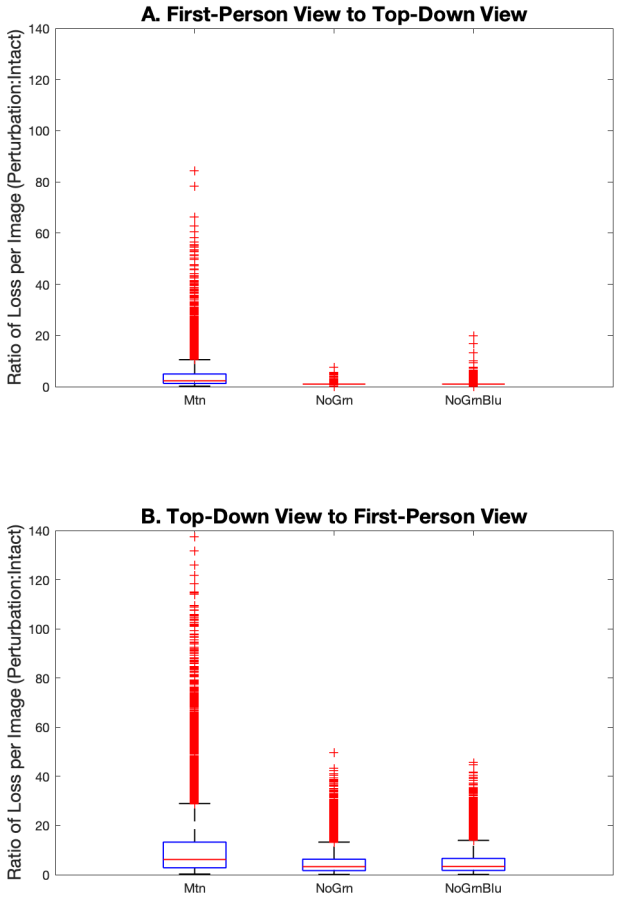


Figure 7.7: Relative loss during perturbation experiments. The figures show the ratio of the loss due to a perturbation loss to the intact model loss for each image. Relative losses are shown for changing the background to mountains (Mtn), removing the green cylinder (NoGrn), and removing both the green and blue cylinders (NoGrnBlu). A. FPV to TDV transformation. B. TDV to FPV transformation. All reconstruction losses for ablations were significantly larger for TDV to FPV than for FPV to TDV transforms.

Perturbing the local and distal cues had three effects. First, the relative loss was significantly greater for the TDV to FPV than the FPV to TDV transformation for both distal and local

cues ($p < 0.0000001$; Wilcoxon sign rank test). In fact, the median loss for FPV to TDV when the green cylinder was zero, meaning that there was no loss in many cases. This makes sense since images in the FPV do not always contain a cylinder. Second, in both transformation directions the loss due to the background change was significantly larger than removing cylinders, and the loss due to removing the green and blue cylinder was significantly greater than removing just the green cylinder. Large loss due to background change makes sense since more pixels in the images are affected. Third, in most cases the transformation losses were relatively small suggesting that with the exception of outliers, the VAEs were able to recover many features in a view image (see Supplemental Figures 25 to 32). Furthermore, at the population level, these perturbations appeared to have a minimal effect on latent variable sensitivity to the distance to an object, heading or place (see Supplemental Figures 33 to 35). The range of correlation values (i.e., many latent variables had strong correlations) and the trends observed in the intact model were preserved.

Perturbations of local and distal cues led to substantial remapping. Table 7.2 shows the percentage of latent variables that were not correlated before the perturbation, but remapped to be significantly correlated with spatial features after the perturbation, and those latent variables that were previously correlated with spatial features prior to the perturbation and remapped to not be correlated with spatial features after the perturbation. There was significantly more remapping when the distal cues changed during FPV to TDV transformations (see top row in Table 7.2), and there tended to be more remapping during FPV to TDV transformations when local cues changed (see bottom two rows in Table 7.2). Such adaptation and remapping has been observed in the retrosplenial cortex [Alexander and Nitz, 2015, Rounds et al., 2018].

Table 7.2: Remapping due to environmental perturbations (LV = 100). Table entries show the % latent variables that became significant ($p < 0.01$) and table entries in parentheses show the % latent variables that became insignificant ($p \geq 0.01$) after the perturbation. Asterisk denotes significantly more remapping for that transformation direction ($p < 0.01$; Wilcoxon rank sum test).

Perturbation	Obj (FPV to TDV)	Obj (TDV to FPV)	Hdg (FPV to TDV)	Hdg (TDV to FPV)	Plc (FPV to TDV)	Plc (TDV to FPV)
Mountains	30%* (18%*)	14% (12%)	28%* (19%*)	8% (10%)	28%* (19%*)	13% (11%)
No green cylinder	2% (2%)	13%* (16%*)	1% (1%)	8% (11%*)	2% (1%)	12%* (15%*)
No green, no blue cylinder	6% (4%)	13%* (17%*)	2% (3%)	8% (11%*)	6% (4%)	13%* (15%*)

7.2.5 Alternative Models

Benefit of Sequences for Linking Views

Unlike the nervous system, the model presented here does not have a temporal component. Rather, a single image from one view is linked to another. We wondered whether a sequence of image views would benefit the ability to link different view perspectives. We created VAEs that took as input a sequence of 5 images from one view and output a reconstruction of the last image in the other view (see Supplemental Figures 38 and 39). Interestingly, the loss was roughly the same for a TDV to a FPV transformation, but the loss was reduced when a sequence of FPVs was used to reconstruct the TDV (Fig. 7.8). This makes intuitive sense because a sequence of FPVs would provide more varied information than a sequence of TDV images. Despite these differences, the reconstruction loss was small in both cases and the spatial metrics in both cases were similar (see Supplemental Figure 41 for spatial metrics and Supplemental Figures 55 through 58 for example place and head direction sensitive latent variables). Although this may be important for a living organism, the computational overhead may outweigh the benefit of using sequences if this model were deployed on a system like a navigating robot.

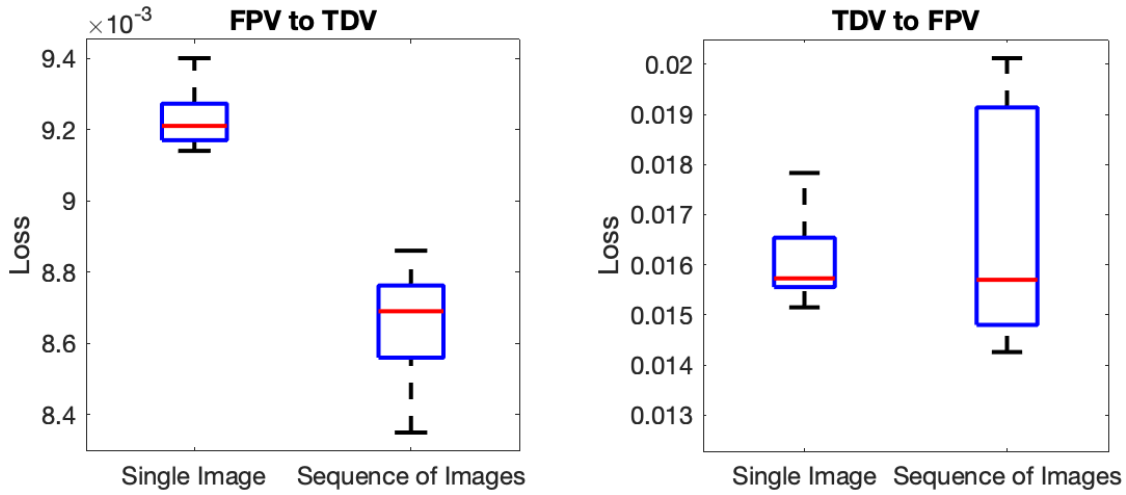


Figure 7.8: Loss comparison between a sequence of images and a single image used for reconstruction.

Combined VAE that Reconstructs FPV to TDV and TDV to FPV Simultaneously

An open issue is whether a single VAE could perform the linkage of views in both directions. We constructed a VAE to test this by interleaving FPVs and TDVs as inputs, while conducting the training for reconstructing the transformed view (see Supplemental Figure 40). This combined model had nearly identical spatial information and coherence as the two separate VAEs (see Supplemental Figure 42). As before, the spatial information and the coherence of latent variables was higher going from a TDV to a FPV than from a FPV to a TDV. Furthermore, many latent variables responded to head direction. Example latent variables that are sensitive to place and heading for the combined model can be seen in Supplemental Figures 59 through 62). It should be noted that although the reconstruction loss going from TPV to FPV was similar, the loss going from FPV to TDV was significantly higher in the combined model (Wilcoxon signed rank test $p < 0.001$). These results suggest that one system could perform this function, but that there may be advantages to having a separate system for each transformation direction.

7.3 Discussion

Using tools from machine learning and artificial intelligence (i.e., Variational AutoEncoders or VAEs), we investigated a fundamental cognitive computation, which is the ability to change one's perspective from a global mental map to an egocentric sensory experience, and vice versa. This is an important, but oftentimes overlooked, aspect of the cognitive map [Tolman, 1948]. Our results suggest that two types of neural activity can support this transformation: 1) Specific representations of locations and objects were observed when reconstructing a first-person view (FPV) from a top-down view (TDV), and 2) Specific representations of heading with more diffuse representations of location were observed when reconstructing a TDV from a FPV. The response of the hidden variables in the present VAEs have similarities to those observed in the brain, with latent variables resembling head direction cells, place cells and cells that encode the distance to objects. Critically, we did not explicitly create such cell types in our model, rather these responses, which resemble place and head direction cells, emerge from the way our model solves this problem.

7.3.1 Neurobiological Evidence for Transformations between Views

Studies with humans and non-human primates have revealed neural correlates for transformations between views or perspectives. In humans, evidence suggests that retrosplenial cortex (RSC) activity is related to route learning from an egocentric viewpoint [Wolbers et al., 2004] and to navigating from a first-person perspective after looking at a top-down map perspective [Sherrill et al., 2013, Zhang et al., 2012]. Although RSC is more active in first-person navigation compared with top-down navigation [Sherrill et al., 2013], the evidence for a transformation is indirect, and there are multiple factors to which the RSC could be responding. Within the first-person perspective, RSC is involved in changing viewpoints to different locations. For example, mentally rotating one's viewpoint to the position of an

avatar or an arrow yields activation in RSC and parietal-occipital sulcus [Lambrey et al., 2012]. In addition, RSC activity is related to the amount of viewpoint change relative to the environmental frame [Sulpizio et al., 2013] and RSC activity is modulated by the magnitude of a viewpoint shift [Sulpizio et al., 2016]. Furthermore, RSC responds to perspective changes when the magnitude of the shift is unknown ahead of time, indicating that it is helpful in making online perspective changes.

Human intracranial recordings in the medial temporal lobe revealed boundary-anchored neural representations that were modulated by one's own as well as another individual's spatial location [Stangl et al., 2021], and recordings of the EC in monkey revealed neurons that represent gaze position in multiple spatial reference frames [Meister and Buffalo, 2018]. These findings more broadly indicate that multiple brain regions in the primate play a role in orienting and processing view-based information from different perspectives [St Jacques et al., 2017, 2018].

In the rat, neurons have been observed that respond to specific spatial frames of reference (e.g., allocentric, egocentric or route-centric), as well as multiple spatial reference frames [Alexander and Nitz, 2015, Jacob et al., 2017, Nitz, 2012]. Some RSC neurons have place specific responses, and the activity of a population of RSC neurons is sufficient to predict the location of a rat in a maze [Alexander and Nitz, 2015]. RSC neurons are sensitive to distance and orientation relative to boundaries [Alexander et al., 2020a]. RSC in both humans [Chrastil et al., 2015, 2016] and rodents [Alexander and Nitz, 2017] has been implicated in mapping distance to other locations in the environment. RSC head direction neurons encode allocentric orientation relative to environmental boundaries [Cho and Sharp, 2001]. RSC activity is sensitive to distance and orientation relative to boundaries and to left versus right turning actions [Alexander et al., 2020b, Alexander and Nitz, 2017]. PPC neurons have been observed to simultaneously map position in multiple external frames of reference. [Nitz, 2012]. Still, none of these studies have put the rodent in situations where it

had multiple viewpoints, which would be difficult to undertake. One study, which is a step in this direction, recorded from the rodent hippocampus and showed place cell responses to itself and to another rat it was observing [Danjo et al., 2018].

However, these studies only involve changes between different first-person viewpoints. Extending this, our simulations suggest a neural solution that uses strong heading signals plus a mixture of place responses to link FPVs to TDVs and more specific place responses with heading to link TDVs to FPVs.

An interesting parallel to the task carried out in our simulations is studies with freely behaving bats. Place cells, head direction cells, and grid cells have been observed in the bat both on the ground when crawling and in the air when flying [Geva-Sagiv et al., 2015, Ulanovsky and Moss, 2007, Sarel et al., 2017]. Similar to [Danjo et al., 2018], social place cells have been found when the bats are viewing other bats [Omer et al., 2018]. GPS tracking of foraging bats over long time periods have demonstrated the ability to use landmarks and take novel routes from a TDV [Toledo et al., 2020, Harten et al., 2020]. Taken together, there is evidence suggesting that encoding and utilizing different spatial perspectives during navigation and memory is a common cognitive function across multiple organisms and multiple brain regions.

7.3.2 Modeling Transformations between Views

Computational neuroscience models have attempted to simulate transformations between allocentric position and orientation in the real world and the egocentric, retina-framed view at that location and orientation. In one influential model, head direction or gaze direction cells modulated activity in RSC by rotating environmental variables [Bicanski and Burgess, 2018, Byrne et al., 2007]. This modulation converted allocentric border or object vector cells into an egocentric bearing to boundaries and objects, and vice versa. Such gain modulated

fields have been postulated and observed in PPC [Pouget and Sejnowski, 1997, Snyder et al., 1998]. In another model, RSC acted as an arbitrator, which depending on the model's confidence in the current task, would activate an allocentric reference frame in the HPC or an egocentric frame in the PPC [Oess et al., 2017]. While these models and others have been useful in suggesting the pathways and neural activity that might produce these transformations, they make assumptions on the underlying computations. For example, Byrne, Becker and Burgess [Byrne et al., 2007] and later Bicanski and Burgess [Bicanski and Burgess, 2018] suggested that the same circuit computed the transformation for both directions. In addition, they did not specifically examine the linkages between FPVs and TDVs.

The present model attempts to be agnostic on how these computations are implemented. Rather than creating a neural network model based on the known responses or connectivity in specific brain regions, we used VAEs to solve the transformation task [Kingma and Welling, 2019], and then tested their feasibility by comparing their responses (i.e., hidden layers and latent variables) to empirical experiments. The latent variables in these VAEs indicate different responses and computations depending on the transformation direction. Furthermore, the separate models for each transformation had less reconstruction loss than a combined model. Whereas strong spatial coding by individual latent variables were observed in TDV to FPV transformations, head direction coding and diffuse place coding were more prevalent in FPV to TDV transformations.

7.3.3 Applying Artificial Neural Networks to Neuroscience

Neuroscientists are turning to Artificial Intelligence (AI) methods to explain their data [Chen et al., 2020a]. For instance, using deep convolutional neural networks (CNNs) as models of hierarchical feature representation in the ventral visual stream can show different cortical

responses in the hidden layers [Cichy and Kaiser, 2019, Güçlü and van Gerven, 2015, Yamins and DiCarlo, 2016]. Moreover, CNNs have shown cortical responses in the dorsal visual stream [Mineault et al., 2021]. Others proposed similar models to synthesize control images to maximally activate specific neuron sites in the monkey V4 [Bashivan et al., 2019]. In a somewhat related robotics study, a deep learning network used the robot's local views and geographic hints, such as satellite images or road maps, to plan paths over a variety of environments [Shah and Levine, 2022]. In the present work, these TDVs were used to predict FPV, and vice versa. This might be an alternative method to localization and mapping in robotics.

The present work compared the sensitivity of latent variables with neural responses. Similarly, latent representations have been used to model the human visual system during working memory tasks [Hedayati et al., 2021]. In another modeling study, a latent factor analysis using dynamical systems was applied to monkey and human motor cortex data to accurately predict behavioral variables and neural dynamics [Pandarinath et al., 2018]. Deep VAEs have been used to interpret fMRI data where there is a lack of labeled data [Qiang et al., 2021]. Our work is another example of how VAEs can be used to model the nervous system and make valuable predictions about the computations and implementations underlying cognitive function.

The present modeling work suggests a means to link a FPV to a TDV and vice versa. Although the modeling work is not based in neurobiology, the different encodings depending on the transformation direction may be compatible with the RSC anatomy [Vann et al., 2009]. Whereas the dysgranular RSC has greater connectivity with cortical regions, such as visual cortex, which provide first-person information [Makino and Komiyama, 2015], granular RSC interacts more with the hippocampal formation and subiculum, which is more sensitive to the allocentric coordinates [Alexander et al., 2020a, Alexander and Nitz, 2017, Olson et al., 2017]. In our simulations, we showed that the model can recover from perturbations, without

retraining, much like place cells in the hippocampus. Moreover, the system did not collapse when large proportions of latent variables were ablated. These perturbation and ablation simulations, suggest that the model can flexibly and rapidly adapt to change, which is a hallmark of neural systems.

In summary, we present a computational model for linking perceptual views, which suggests a potential neural implementation for this cognitive function. Furthermore, it makes predictions regarding the functional anatomy suggesting separate encodings depending on the direction of the view transformation, and the ability to adapt without retraining when challenged with perturbations. Although this model provides a possible implementation, we do not yet know exactly how the mammalian brain carries out such a task. Therefore, it will be of interest to follow up this modeling study with similar experiments tailored for humans and other animals. Furthermore, linking different views, as in [Shah and Levine, 2022], may be applicable to robot navigation.

7.4 Materials and Methods

7.4.1 Robot Simulations

The Webots robot environment [Webots, Michel, 2004] was used to simulate an animal freely exploring its environment (Fig. 7.1). The Khepera robot, which is a two-wheeled robot produced by K-Team, was used for the simulations. During exploration, the robot had a 50% chance of moving straight, 25% chance of veering (i.e., an arcing turn) toward the left, and 25% chance of veering to the right. The robot has 8 distance sensors that were used to detect the arena walls and the cylinders. If detected, the robot rotated, with a 50% chance, either clockwise or counterclockwise until the front facing distance detectors were clear. A camera was mounted on top of the robot for the FPV. Every update cycle,

the camera frame was converted into a 64x64 RGB image (FPV) and a simulated overhead camera took a JPEG image (TDV) of the robot in its environment. During the exploration, the TDV from the simulator, the FPV from the robot's camera, and other environmental parameters (e.g., place, heading, distance to object) were collected and saved. The "entrance hall" was used as a default background. In the perturbation experiments, this was replaced with the "mountains" background which was a desert scene with mountains in the distance. During the local cue perturbation experiments, either the green cylinder or both the green and blue cylinders were rendered invisible using the transparency setting in Webots. The Khepera's distance sensors still detected the object, but they were not visible by the camera. The same random number generator seed was used on all simulation runs to ensure that the robot's trajectory was the same in each condition. The software used to run the simulation will be made available on GitHub upon publication.

7.4.2 Variational Autoencoder Construction and Latent Variable Analysis

VAEs [Kingma and Welling, 2019] were constructed to transform between TDV and FPV. Briefly, the VAE design is as follows. The perspective transformation model used in the preliminary results is based on standard VAE training whose loss includes a reconstruction loss term and a KL divergence term. The reconstruction term optimizes the network so that the input could be reconstructed while the KL divergence term is used to constrain the latent representation close to the prior distribution. To promote stable training, we used KL annealing to gradually increase the weight of the KL term from 0 to 1 [Bowman et al., 2016]. The model was trained for 20,000 epochs. In the first 50 epochs, the KL term increased linearly from 0 to 1. After 50 epochs, the KL term weight was kept at 1. After the VAE was trained, TDVs or FPVs were presented to the model. We then can measure the latent variable sensitivity by examining how much each latent variable changes with environmental

changes. More details are given in the supplemental materials.

The VAE's latent variables were analyzed for sensitivity to objects, heading, and place.

Object sensitivity was measured by Pearson's correlation of the latent variable to the distance from the robot to the red, green and blue cylinder. The distance function was given by eqn. 7.1:

$$dCyl_{ti} = \sum_{t=1}^{T} \sum_{i=1}^{3} \|(loc_t - cyl_i)\| \tag{7.1}$$

Where $\|(loc_t - cyl_i)\|$, is the Euclidean distance between the location of the robot, loc, and the location of cylinder, cyl. The distance, $dCyl$, was calculated for each i cylinder at time t with T equal to 2000 time steps. This created a vector of length 2000 of the distances to each cylinder object ($dCyl$) in the simulation. The sensitivity of each latent variable to the cylinder objects was then given by eqn. 7.2:

$$cyl_{ni} = \sum_{n=1}^{N} \sum_{i=1}^{3} corr(lv_n, dCyl_i) \tag{7.2}$$

Where N is the number of latent variables, i is the preferred direction, lv_n is the response of the latent variable n for the 2000 time steps. The resulting cyl_{ni} are correlation coefficients for each latent variable to the red, green, and blue cylinder objects.

Head direction sensitivity was measured by Pearson's correlation of the latent variable to a cosine tuning curve with one of 16 preferred directions, which were evenly spaced from 0 to 2π. The cosine tuning curve was given by (eqn. 7.3):

$$rHD_{ti} = \sum_{t=1}^{T} \sum_{i=1}^{16} max(0.0, cos(rot_t - pd_i)) \tag{7.3}$$

Where the expected cosine tuning response for each i preferred direction pd was calculated based on the robot's heading rot at data point t with T equal to 2000 time steps. This created a vector of length 2000 of expected head direction responses for each preferred direction (rHD). The sensitivity of each latent variable to head direction was then given by eqn. 7.4:

$$hd_{ni} = \sum_{n=1}^{N} \sum_{i=1}^{16} corr(lv_n, rHD_i) \tag{7.4}$$

Where N is the number of latent variables, i is the preferred direction index, and lv_n is the response of the latent variable n for the 2000 time steps. The resulting hd_{ni} are correlation coefficients for each latent variable for each of the 16 idealized head direction cells.

Place cell sensitivity was measured by Pearson's correlation of the latent variable to a 2-dimensional Gaussian centered at one of 16 locations, which were evenly spaced across the robot's arena. The Gaussian function was given by eqn. 7.5:

$$rPlc_{ti} = \sum_{t=1}^{T} \sum_{i=1}^{16} \frac{exp(-\|(loc_t - ctr_i)\|)}{\sigma} \tag{7.5}$$

Where $\|(loc_t - ctr_i)\|$, is the Euclidean distance between the location of the robot, loc, and the centroid of the place cell, ctr, and σ was set to 0.33. The response, $rPlc$, was calculated

for each i place at time t with T equal to 2000 time steps. This created a vector of length 2000 of expected place cell responses for each location ($rPlc$). The sensitivity of each latent variable to place was then given by eqn. 7.6:

$$plc_{ni} = \sum_{n=1}^{N} \sum_{i=1}^{16} corr(lv_n, rPlc_i) \tag{7.6}$$

Where N is the number of latent variables, i is the preferred direction, lv_n is the response of the latent variable n for the 2000 time steps. The resulting plc_{ni} are correlation coefficients for each latent variable for each of the 16 idealized place cells.

Chapter 8

Conclusions

8.1 Summary

In recent years, there has been a growing convergence between machine learning and neuroscience. On the one hand, machine learning could benefit from the insights and inspiration provided by the discoveries in neuroscience, as well as the integration of biologically-inspired components. On the other hand, machine learning techniques can also be used to enhance our knowledge of the brain and its functions.

The projects introduced in this dissertation all involve machine learning techniques and neuroscience findings. For the projects of neuromodulated patience control for robot navigation (Chapter 3) and reinforcement learning with neuromodulation systems for dynamic environment adaptation (Chapter 4), we integrated 5-HT, NE and ACh neuromodulation systems to help the self-driving mobile robot and reinforcement learning agents to achieve better performance. Inspired by the latent representation in brain, we proposed a method called LUSR to improve the domain adaptation performance of reinforcement learning algorithms by addressing the adaptation problem from the pixel domain to a latent space (Chapter

5). Furthermore, inspired by memory consolidation in the cortex, we introduced a method of policy distillation with selective input gradient regularization to achieve both computation efficiency and high interpretability for explainable reinforcement learning (Chapter 6). These chapters showed how inspiration from neuroscience could improve machine learning. In contrast to the other chapters, , my work of linking first-person views and global views utilized the machine learning technique of VAE to enhance our understanding about how brain may conduct the view transformation in a 3-D environment (Chapter 7).

Overall, this dissertation demonstrated the mutual beneficial relationship between the fields of machine learning and neuroscience. The successful integration of machine learning and brain mechanisms/findings in the projects encourage more developments of bio-inspired machine learning and AI-augmented neuroscience research.

8.2 Future Directions

The bio-inspired machine learning projects in this defense focus on the utilization of bio-inspired components (e.g. neuromodulation systems) and high-level intuitions (e.g. memory consolidation) to help the machine learning system. One limitation of projects introduced in Chapter 3 and 4 is that the machine learning and bio-inspired components are trained or updated separately. For example, in the project of utilizing neuromodulation systems to help reinforcement learning agents to adapt to dynamic environment changes, the training of reinforcement learning policies and the update of NE and ACh neuromodulation systems were conducted separately. To further utilize neuroscience inspirations, one future direction is integrating them into the learning of machine learning techniques. For example, dopamine neuromodulation is a vital mechanism in the brain that regulates reward and dopamine neurons play a key role in reinforcement learning. We may try to integrate dopamine neuromodulation into the learning method of reinforcement learning.

A potential area of future research involves expanding upon the project presented in Chapter 5, which deals with the problem of domain adaptation in reinforcement learning. In Chapter 5, the assumption was made that the test domains are visually distinct from the training domain, but share the same underlying dynamics. This assumption restricts the applicability of our approach. To overcome this limitation, we could broaden the problem to encompass transfer learning. In doing so, the test domains would not be required to share the same underlying dynamics as the training domain. However, this approach raises a broader question of how to effectively transfer the knowledge acquired from the policy trained in the original domain to the new domains. One possible solution to this question is to use the technique of policy distillation, which was introduced in Chapter 6. However, further research is needed to fully address this issue.

Future research could also build upon the perspective transformation project, which utilized a standard variational autoencoder (VAE) to learn latent encodings within a simulated robot environment. To advance this work, the experiment could be upgraded from a simulated robot to a real mobile robot navigating in an outdoor park. This would require the use of more advanced deep learning techniques to handle the complex real-world environment. For example, advanced attention mechanisms, transformer architectures, or diffusion models could be induced into the neural network models. Such improvements in the experiment setting and deep learning techniques may lead to further advancements in the findings.

Bibliography

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

A. S. Alexander and D. A. Nitz. Retrosplenial cortex maps the conjunction of internal and external spaces. *Nat Neurosci*, 18(8):1143–51, 2015. ISSN 1546-1726 (Electronic) 1097-6256 (Linking). doi: 10.1038/nn.4058.

A. S. Alexander and D. A. Nitz. Spatially periodic activation patterns of retrosplenial cortex encode route sub-spaces and distance traveled. *Curr Biol*, 27(11):1551–1560 e4, 2017. ISSN 1879-0445 (Electronic) 0960-9822 (Linking). doi: 10.1016/j.cub.2017.04.036.

A. S. Alexander, L. C. Carstensen, J. R. Hinman, F. Raudies, G. W. Chapman, and M. E. Hasselmo. Egocentric boundary vector tuning of the retrosplenial cortex. *Sci Adv*, 6(8): eaaz2322, 2020a. ISSN 2375-2548 (Electronic) 2375-2548 (Linking). doi: 10.1126/sciadv.aaz2322.

A. S. Alexander, J. C. Robinson, H. Dannenberg, N. R. Kinsky, S. J. Levy, W. Mau, G. W. Chapman, D. W. Sullivan, and M. E. Hasselmo. Neurophysiological coding of space and time in the hippocampus, entorhinal cortex, and retrosplenial cortex. *Brain Neurosci Adv*, 4:2398212820972871, 2020b. ISSN 2398-2128 (Electronic) 2398-2128 (Linking). doi: 10.1177/2398212820972871.

Mariam Aly and Nicholas B Turk-Browne. How hippocampal memory shapes, and is shaped by, attention. *The hippocampus from cells to systems: structure, connectivity, and functional contributions to memory and flexible cognition*, pages 369–403, 2017.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkl3m1BFDB`.

Michael C Avery and Jeffrey L Krichmar. Neuromodulatory systems and their interactions: a review of models, theories, and experiments. *Frontiers in neural circuits*, page 108, 2017.

P. Bashivan, K. Kar, and J. J. DiCarlo. Neural population control via deep image synthesis. *Science*, 364(6439), 2019. ISSN 1095-9203 (Electronic) 0036-8075 (Linking). doi: 10.1126/science.aav9436.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

A. Bicanski and N. Burgess. A neural-level model of spatial memory and imagery. *Elife*, 7, 2018. ISSN 2050-084X (Electronic) 2050-084X (Linking). doi: 10.7554/eLife.33752.

Narcisse P Bichot, Matthew T Heard, Ellen M DeGennaro, and Robert Desimone. A source for feature-based attention in the prefrontal cortex. *Neuron*, 88(4):832–844, 2015.

Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Larry J Ackel, Urs Muller, Phil Yeres, and Karol Zieba. Visualbackprop: Efficient visualization of cnns for autonomous driving. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4701–4708. IEEE, 2018.

Sebastien Bouret and Susan J Sara. Network reset: a simplified overarching theory of locus coeruleus noradrenaline function. *Trends in neurosciences*, 28(11):574–582, 2005.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 10–21. Association for Computational Linguistics, 2016. doi: 10.18653/v1/K16-1002.

Amaury Bréhéret. Pixel annotation tool. `https://github.com/abreheret/PixelAnnotationTool`, 2017.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

P. Byrne, S. Becker, and N. Burgess. Remembering the past and imagining the future: a neural model of spatial memory and imagery. *Psychol Rev*, 114(2):340–75, 2007. ISSN 0033-295X (Print) 0033-295X (Linking). doi: 10.1037/0033-295X.114.2.340.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. Model-free deep reinforcement learning for urban autonomous driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2765–2771. IEEE, 2019.

Kexin Chen, Tiffany Hwu, Hirak J. Kashyap, Jeffrey L. Krichmar, Kenneth Stewart, Jinwei Xing, and Xinyun Zou. Neurorobots as a means toward neuroethology and explainable ai. *Frontiers in Neurorobotics*, 14(70), 2020a. ISSN 1662-5218. doi: 10.3389/fnbot.2020.570308.

Kexin Chen, Tiffany Hwu, Hirak J Kashyap, Jeffrey L Krichmar, Kenneth Stewart, Jinwei Xing, and Xinyun Zou. Neurorobots as a means toward neuroethology and explainable ai. *Frontiers in Neurorobotics*, 14:570308, 2020b.

Nicholas Cheney, Martin Schrimpf, and Gabriel Kreiman. On the robustness of convolutional neural networks to internal architecture and weight perturbations. *arXiv preprint arXiv:1703.08245*, 2017.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

J. Cho and P. E. Sharp. Head direction, place, and movement correlates for cells in the rat retrosplenial cortex. *Behav Neurosci*, 115(1):3–25, 2001. ISSN 0735-7044 (Print) 0735-7044 (Linking). doi: 10.1037/0735-7044.115.1.3.

E. R. Chrastil, K. R. Sherrill, M. E. Hasselmo, and C. E. Stern. There and back again: Hippocampus and retrosplenial cortex track homing distance during human path integration. *J Neurosci*, 35(46):15442–52, 2015. ISSN 1529-2401 (Electronic) 0270-6474 (Linking). doi: 10.1523/JNEUROSCI.1209-15.2015.

E. R. Chrastil, K. R. Sherrill, M. E. Hasselmo, and C. E. Stern. Which way and how far? tracking of translation and rotation information for human path integration. *Hum Brain Mapp*, 37(10):3636–55, 2016. ISSN 1097-0193 (Electronic) 1065-9471 (Linking). doi: 10.1002/hbm.23265.

E. R. Chrastil, S. M. Tobyne, R. K. Nauer, A. E. Chang, and C. E. Stern. Converging meta-analytic and connectomic evidence for functional subregions within the human retrosplenial region. *Behav Neurosci*, 132(5):339–355, 2018. ISSN 1939-0084 (Electronic) 0735-7044 (Linking). doi: 10.1037/bne0000278. URL `https://www.ncbi.nlm.nih.gov/pubmed/30321025`.

R. M. Cichy and D. Kaiser. Deep neural networks as scientific models. *Trends Cogn Sci*, 23 (4):305–317, 2019. ISSN 1879-307X (Electronic) 1364-6613 (Linking). doi: 10.1016/j.tics.2019.01.009.

B. J. Clark, C. M. Simmons, L. E. Berkowitz, and A. A. Wilber. The retrosplenial-parietal network and reference frame coordination for spatial navigation. *Behav Neurosci*, 132(5):416–429, 2018. ISSN 1939-0084 (Electronic) 0735-7044 (Linking). doi: 10.1037/bne0000260.

Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1331–1340. PMLR, 2019.

T. Danjo, T. Toyoizumi, and S. Fujisawa. Spatial representations of self and other in the hippocampus. *Science*, 359(6372):213–218, 2018. ISSN 1095-9203 (Electronic) 0036-8075 (Linking). doi: 10.1126/science.aao3898.

Giang Dao, Indrajeet Mishra, and Minwoo Lee. Deep reinforcement learning monitor for snapshot recording. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 591–598. IEEE, 2018.

Giang Dao, Wesley Houston Huff, and Minwoo Lee. Learning sparse evidence-driven interpretation to understand deep reinforcement learning agents. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2021.

D. Derdikman and E. I. Moser. A manifold of spatial maps in the brain. *Trends Cogn Sci*, 14 (12):561–9, 2010. ISSN 1879-307X (Electronic) 1364-6613 (Linking). doi: 10.1016/j.tics.2010.09.004.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017a.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017b.

R. A. Epstein, E. Z. Patai, J. B. Julian, and H. J. Spiers. The cognitive map in humans: spatial navigation and beyond. *Nat Neurosci*, 20(11):1504–1513, 2017. ISSN 1546-1726 (Electronic) 1097-6256 (Linking). doi: 10.1038/nn.4656.

U. M. Erdem and M. Hasselmo. A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *Eur. J. Neurosci.*, 35:916–931, 2012.

Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, and James Davidson. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5113–5120. IEEE, 2018.

Aidin Ferdowsi, Ursula Challita, Walid Saad, and Narayan B Mandayam. Robust deep reinforcement learning for security and safety in autonomous vehicle systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 307–312. IEEE, 2018.

Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

Loren M. Frank, Garrett B. Stanley, and Emery N. Brown. Hippocampal plasticity across multiple days of exposure to novel environments. *Journal of Neuroscience*, 24(35):7681–7689, 2004. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.1958-04.2004.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *International Conference on Machine Learning*, pages 2063–2072, 2019.

Philippe Gaussier, Jean Paul Banquet, Nicolas Cuperlier, Mathias Quoy, Lise Aubin, Pierre-Yves Jacob, Francesca Sargolini, Etienne Save, Jeffrey L Krichmar, and Bruno Poucet. Merging information in the entorhinal cortex: what can we learn from robotics experiments and modeling? *Journal of Experimental Biology*, 222(Suppl_1):jeb186932, 2019.

M. Geva-Sagiv, L. Las, Y. Yovel, and N. Ulanovsky. Spatial cognition in bats and rats: from sensory acquisition to multiscale maps and navigation. *Nat Rev Neurosci*, 16(2):94–108, 2015. ISSN 1471-0048 (Electronic) 1471-003X (Linking). doi: 10.1038/nrn3888.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Stephanie L Grella, Jonathan M Neil, Hilary T Edison, Vanessa D Strong, Irina V Odintsova, Susan G Walling, Gerard M Martin, Diano F Marrone, and Carolyn W Harley. Locus coeruleus phasic, but not tonic, activation initiates global remapping in a familiar environment. *Journal of Neuroscience*, 39(3):445–455, 2019.

Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pages 1792–1801, 2018.

Umut Güçlü and Marcel A. J. van Gerven. Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *The Journal of Neuroscience*, 35(27):10005–10014, 2015. doi: 10.1523/JNEUROSCI.5023-14.2015.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

Todd A Hare, John O'doherty, Colin F Camerer, Wolfram Schultz, and Antonio Rangel. Dissociating the role of the orbitofrontal cortex and the striatum in the computation of goal values and prediction errors. *Journal of neuroscience*, 28(22):5623–5630, 2008.

P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern*, 4:100–107, 1968.

Lee Harten, Amitay Katz, Aya Goldshtein, Michal Handel, and Yossi Yovel. The ontogeny of a mammalian cognitive map in the real world. *Science*, 369(6500):194, 2020. doi: 10.1126/science.aay3354.

Michael E Hasselmo and Jill McGaughy. High acetylcholine levels set circuit dynamics for attention and encoding and low acetylcholine levels set dynamics for consolidation. *Progress in brain research*, 145:207–231, 2004.

Shekoofeh Hedayati, Ryan O'Donnell, and Brad Wyble. Memory for latent representations: An account of working memory that builds on visual knowledge for efficient and detailed visual representations. *bioRxiv*, 2021.02.07.430171:1–59, 2021. doi: 10.1101/2021.02.07.430171.

Irina Higgins, Arka Pal, Andrei A Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:1707.08475*, 2017.

Zhang-Wei Hong, Chen Yu-Ming, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, Hsuan-Kung Yang, Brian Hsi-Lin Ho, Chih-Chieh Tu, Yueh-Chuan Chang, Tsu-Ching Hsiao, et al. Virtual-to-real: Learning to control in visual semantic segmentation. *arXiv preprint arXiv:1802.00285*, 2018.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Tiffany Hwu, Alexander Y Wang, Nicolas Oros, and Jeffrey L Krichmar. Adaptive robot path planning using a spiking neuron algorithm with axonal delays. *IEEE Transactions on Cognitive and Developmental Systems*, 10(2):126–137, 2017.

Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. On relating explanations and adversarial examples. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/7392ea4ca76ad2fb4c9c3b6a5c6e31e3-Paper.pdf`.

Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara. Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 144–150, 2018.

P. Y. Jacob, G. Casali, L. Spieser, H. Page, D. Overington, and K. Jeffery. An independent, landmark-dominated head-direction signal in dysgranular retrosplenial cortex. *Nat Neurosci*, 20(2):173–175, 2017. ISSN 1546-1726 (Electronic) 1097-6256 (Linking). doi: 10.1038/nn.4465.

Ananya Harsh Jha, Saket Anand, Maneesh Singh, and VSR Veeravasarapu. Disentangling factors of variation with cycle-consistent variational auto-encoders. In *European Conference on Computer Vision*, pages 829–845. Springer, 2018.

Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

Gregory Kahn, Pieter Abbeel, and Sergey Levine. Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, 6(2):1312–1319, 2021.

S. M. Kim, S. Ganguli, and L. M. Frank. Spatial information outflow from the hippocampal circuit: distributed spatial coding and phase precession in the subiculum. *J Neurosci*, 32(34):11539–58, 2012. ISSN 1529-2401 (Electronic) 0270-6474 (Linking). doi: 10.1523/JNEUROSCI.5942-11.2012.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019. ISSN 1935-8237. doi: 10.1561/2200000056.

J. L. Kubie, R. U. Muller, and E. Bostock. Spatial firing properties of hippocampal theta cells. *J Neurosci*, 10(4):1110–23, 1990. ISSN 0270-6474 (Print) 0270-6474 (Linking).

S. Lambrey, C. Doeller, A. Berthoz, and N. Burgess. Imagining being somewhere else: neural basis of changing perspective in space. *Cereb Cortex*, 22(1):166–74, 2012. ISSN 1460-2199 (Electronic) 1047-3211 (Linking). doi: 10.1093/cercor/bhr101.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020a.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020b.

S.M. Lavalle. Motion planning: Part ii: Wild frontiers. *IEEE Robot. Autom. Mag.*, 18: 108–118, 2011.

Ngan Le, Vidhiwar Singh Rathour, Kashu Yamazaki, Khoa Luu, and Marios Savvides. Deep reinforcement learning in computer vision: a comprehensive survey. *Artificial Intelligence Review*, pages 1–87, 2021.

Hongliang Li, Derong Liu, and Ding Wang. Manifold regularized reinforcement learning. *IEEE Transactions on neural networks and Learning Systems*, 29(4):932–943, 2017.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Zhengxian Lin, Kim-Ho Lam, and Alan Fern. Contrastive explanations for reinforcement learning via embedded self predictions. *arXiv preprint arXiv:2010.05180*, 2020.

Guiliang Liu, Xiangyu Sun, Oliver Schulte, and Pascal Poupart. Learning tree interpretation from object representation for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:19622–19636, 2021.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`.

H. Makino and T. Komiyama. Learning enhances the relative impact of top-down processing in the visual cortex. *Nat Neurosci*, 18(8):1116–22, 2015. ISSN 1546-1726 (Electronic) 1097-6256 (Linking). doi: 10.1038/nn.4061.

Rowan McAllister, Gregory Kahn, Jeff Clune, and Sergey Levine. Robustness to out-of-distribution inputs via task-aware generative uncertainty. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2083–2089. IEEE, 2019.

M. Meister. Memory system neurons represent gaze position and the visual world. *J Exp Neurosci*, 12:1179069518787484, 2018. ISSN 1179-0695 (Print) 1179-0695 (Linking). doi: 10.1177/1179069518787484.

M. L. R. Meister and E. A. Buffalo. Neurons in primate entorhinal cortex represent gaze position in multiple spatial reference frames. *J Neurosci*, 38(10):2430–2441, 2018. ISSN 1529-2401 (Electronic) 0270-6474 (Linking). doi: 10.1523/JNEUROSCI.2432-17.2018.

Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004. URL `http://www.ars-journal.com/International-Journal-of-Advanced-Robotic-Systems/Volume-1/39-42.pdf`.

Michael Milford and Ruth Schulz. Principles of goal-directed spatial robot navigation in biomimetic models. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1655):20130484, 2014.

Earl K Miller and Timothy J Buschman. Neural mechanisms for the executive control of attention. 2014.

Patrick J Mineault, Shahab Bhaktiari, Blake A Richards, and Christopher C Pack. Your head is there to move you around: Goal-driven models of the primate dorsal pathway. *bioRxiv*, DOI: 10.1101/2021.07.09.451701, 2021. doi: 10.1101/2021.07.09.451701. URL `https://www.biorxiv.org/content/early/2021/07/19/2021.07.09.451701`.

Katsuhiko Miyazaki, Kayoko W Miyazaki, Akihiro Yamanaka, Tomoki Tokuda, Kenji F Tanaka, and Kenji Doya. Reward probability and timing uncertainty alter the effect of dorsal raphe serotonin neurons on patience. *Nature communications*, 9(1):2048, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

P Read Montague, Peter Dayan, and Terrence J Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of neuroscience*, 16(5): 1936–1947, 1996.

Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/e9510081ac30ffa83f10b68cde1cac07-Paper.pdf`.

Takashi Nagata, Jinwei Xing, Tsutomu Kumazawa, and Emre Neftci. Uncertainty aware model integration on reinforcement learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2022.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Understanding neural networks via feature visualization: A survey. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 55–76. Springer, 2019.

D. A. Nitz. Spaces within spaces: rat parietal cortex neurons register position across three reference frames. *Nat Neurosci*, 15(10):1365–7, 2012. ISSN 1546-1726 (Electronic) 1097-6256 (Linking). doi: 10.1038/nn.3213.

T. Oess, J. L. Krichmar, and F. Rohrbein. A computational model for spatial navigation based on reference frames in the hippocampus, retrosplenial cortex, and posterior parietal cortex. *Frontiers in Neurorobotics*, 11, 2017. ISSN 1662-5218. doi: 10.3389/fnbot.2017.00004.

Barry S Oken, Martin C Salinsky, and SM2865224 Elsas. Vigilance, alertness, or sustained attention: physiological basis and measurement. *Clinical neurophysiology*, 117(9):1885–1901, 2006.

J. M. Olson, K. Tongprasearth, and D. A. Nitz. Subiculum neurons map the current axis of travel. *Nat Neurosci*, 20(2):170–172, 2017. ISSN 1546-1726 (Electronic) 1097-6256 (Linking). doi: 10.1038/nn.4464.

D. B. Omer, S. R. Maimon, L. Las, and N. Ulanovsky. Social place-cells in the bat hippocampus. *Science*, 359(6372):218–224, 2018. ISSN 1095-9203 (Electronic) 0036-8075 (Linking). doi: 10.1126/science.aao3474.

Xinlei Pan, Yurong You, Ziyan Wang, and Cewu Lu. Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*, 2017.

Chethan Pandarinath, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, Jaimie M. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, 2018. ISSN 1548-7105. doi: 10.1038/s41592-018-0109-9.

Nathan F Parker, Courtney M Cameron, Joshua P Taliaferro, Junuk Lee, Jung Yoon Choi, Thomas J Davidson, Nathaniel D Daw, and Ilana B Witten. Reward and choice encoding in terminals of midbrain dopamine neurons depends on striatal target. *Nature neuroscience*, 19(6):845–854, 2016.

Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

Michael I Posner. Measuring alertness. *Annals of the New York Academy of Sciences*, 1129 (1):193–199, 2008.

A. Pouget and T. J. Sejnowski. Spatial transformations in the parietal cortex using basis functions. *J Cogn Neurosci*, 9(2):222–37, 1997. ISSN 0898-929X (Print) 0898-929X (Linking). doi: 10.1162/jocn.1997.9.2.222.

Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SJgzLkBKPB`.

N. Qiang, Q. Dong, F. Ge, H. Liang, B. Ge, S. Zhang, Y. Sun, J. Gao, and T. Liu. Deep variational autoencoder for mapping functional brain networks. *IEEE Transactions on Cognitive and Developmental Systems*, 13(4):841–852, 2021. ISSN 2379-8939. doi: 10.1109/TCDS.2020.3025137.

Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Matthias Rosynski, Frank Kirchner, and Matias Valdenegro-Toro. Are gradient-based saliency maps useful in deep reinforcement learning? In *"I Can't Believe It's Not Better!" NeurIPS 2020 workshop*, 2020. URL `https://openreview.net/forum?id=ZF4KyC2zz6x`.

E. L. Rounds, A. S. Alexander, D. A. Nitz, and J. L. Krichmar. Conjunctive coding in an evolved spiking model of retrosplenial cortex. *Behavioral Neuroscience*, 132(5):430–452, 2018. ISSN 0735-7044. doi: 10.1037/bne0000236.

Christian Rupprecht, Cyril Ibrahim, and Christopher J Pal. Finding and visualizing weaknesses of deep reinforcement learning agents. *arXiv preprint arXiv:1904.01318*, 2019.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

A. Sarel, A. Finkelstein, L. Las, and N. Ulanovsky. Vectorial representation of spatial goals in the hippocampus of bats. *Science*, 355(6321):176–180, 2017. ISSN 1095-9203 (Electronic) 0036-8075 (Linking). doi: 10.1126/science.aak9589.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Dhruv Shah and Sergey Levine. Viking: Vision-based kilometer-scale navigation with geographic hints. *arXiv*, 2202.11271 [cs.RO], 2022.

P. E. Sharp. Subicular cells generate similar spatial firing patterns in two geometrically and visually distinctive environments: comparison with hippocampal place cells. *Behav Brain Res*, 85(1):71–92, 1997. ISSN 0166-4328 (Print) 0166-4328 (Linking). doi: 10.1016/s0166-4328(96)00165-9.

Katherine R. Sherrill, Uğur M. Erdem, Robert S. Ross, Thackery I. Brown, Michael E. Hasselmo, and Chantal E. Stern. Hippocampus and retrosplenial cortex combine path integration signals for successful navigation. *The Journal of Neuroscience*, 33(49):19304, 2013. doi: 10.1523/JNEUROSCI.1825-13.2013.

Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International conference on artificial intelligence and statistics*, pages 1855–1865. PMLR, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

W. E. Skaggs, B. L. McNaughton, M. A. Wilson, and C. A. Barnes. Theta phase precession in hippocampal neuronal populations and the compression of temporal sequences. *Hippocampus*, 6(2):149–72, 1996. ISSN 1050-9631 (Print) 1050-9631 (Linking). doi: 10.1002/(SICI)1098-1063(1996)6:2⟨149::AID-HIPO6⟩3.0.CO;2-K.

Reda Bahi Slaoui, William R. Clements, Jakob N. Foerster, and Sébastien Toth. Robust domain randomization for reinforcement learning, 2020. URL `https://openreview.net/forum?id=H1xSOTVtvH`.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

L. H. Snyder, K. L. Grieve, P. Brotchie, and R. A. Andersen. Separate body- and world-referenced representations of visual space in parietal cortex. *Nature*, 394(6696):887–91, 1998. ISSN 0028-0836 (Print) 0028-0836 (Linking). doi: 10.1038/29777.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

P. L. St Jacques, K. K. Szpunar, and D. L. Schacter. Shifting visual perspective during retrieval shapes autobiographical memories. *Neuroimage*, 148:103–114, 2017. ISSN 1095-9572 (Electronic) 1053-8119 (Linking). doi: 10.1016/j.neuroimage.2016.12.028.

P. L. St Jacques, A. C. Carpenter, K. K. Szpunar, and D. L. Schacter. Remembering and imagining alternative versions of the personal past. *Neuropsychologia*, 110:170–179, 2018. ISSN 1873-3514 (Electronic) 0028-3932 (Linking). doi: 10.1016/j.neuropsychologia.2017.06.015.

M. Stangl, U. Topalovic, C. S. Inman, S. Hiller, D. Villaroman, Z. M. Aghajan, L. Christov-Moore, N. R. Hasulak, V. R. Rao, C. H. Halpern, D. Eliashiv, I. Fried, and N. Suthana. Boundary-anchored neural mechanisms of location-encoding for self and others. *Nature*, 589(7842):420–425, 2021. ISSN 1476-4687 (Electronic) 0028-0836 (Linking). doi: 10.1038/s41586-020-03073-y.

A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3310–3317 vol.4, May 1994. doi: 10.1109/ROBOT.1994.351061.

V. Sulpizio, G. Committeri, S. Lambrey, A. Berthoz, and G. Galati. Selective role of lingual/parahippocampal gyrus and retrosplenial complex in spatial memory across viewpoint changes relative to the environmental reference frame. *Behav Brain Res*, 242:62–75, 2013. ISSN 1872-7549 (Electronic) 0166-4328 (Linking). doi: 10.1016/j.bbr.2012.12.031.

V. Sulpizio, G. Committeri, S. Lambrey, A. Berthoz, and G. Galati. Role of the human retrosplenial cortex/parieto-occipital sulcus in perspective priming. *Neuroimage*, 125:108–119, 2016. ISSN 1095-9572 (Electronic) 1053-8119 (Linking). doi: 10.1016/j.neuroimage.2015.10.040.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

Guanhong Tao, Shiqing Ma, Yingqi Liu, and Xiangyu Zhang. Attacks meet interpretability: Attribute-steered detection of adversarial samples. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/b994697479c5716eda77e8e9713e5f0f-Paper.pdf`.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

Sivan Toledo, David Shohami, Ingo Schiffner, Emmanuel Lourie, Yotam Orchan, Yoav Bartan, and Ran Nathan. Cognitive map–based navigation in wild bats revealed by a new high-throughput tracking system. *Science*, 369(6500):188, 2020. doi: 10.1126/science.aax6904.

E. C. Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948. ISSN 0033-295X. doi: Doi10.1037/H0061626.

Nicholay Topin and Manuela Veloso. Generation of policy-level explanations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2514–2521, 2019.

Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Pieter Abbeel, Sergey Levine, Kate Saenko, and Trevor Darrell. Adapting deep visuomotor representations with weak pairwise constraints. In *Algorithmic Foundations of Robotics XII*, pages 688–703. Springer, 2020.

N. Ulanovsky and C. F. Moss. Hippocampal cellular and network activity in freely moving echolocating bats. *Nat Neurosci*, 10(2):224–33, 2007. ISSN 1097-6256 (Print) 1097-6256 (Linking). doi: 10.1038/nn1829.

Christopher Urmson, Reid Simmons, and Issa Nesnas. A generic framework for robotic navigation. In *Proceedings of the IEEE Aerospace Conference*, volume 5, pages 2463–2470. Citeseer, 2003.

S. D. Vann, J. P. Aggleton, and E. A. Maguire. What does the retrosplenial cortex do? *Nat Rev Neurosci*, 10(11):792–802, 2009. ISSN 1471-0048 (Electronic) 1471-003X (Linking). doi: 10.1038/nrn2733.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR, 2018.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11 (12), 2010.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Ding Wang, Mingming Ha, and Mingming Zhao. The intelligent critic framework for advanced optimal control. *Artificial Intelligence Review*, 55(1):1–22, 2022.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International Conference on Machine Learning*, pages 9940–9951. PMLR, 2020.

Yang Wang, David Mulvaney, Ian Sillitoe, and Erick Swere. Robot navigation by waypoints. *Journal of Intelligent and Robotic Systems*, 52:175–207, 2008.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

Webots. http://www.cyberbotics.com. URL `http://www.cyberbotics.com`. Open-source Mobile Robot Simulation Software.

A. A. Wilber, B. J. Clark, A. J. Demecha, L. Mesina, J. M. Vos, and B. L. McNaughton. Cortical connectivity maps reveal anatomically distinct areas in the parietal cortex of the rat. *Front Neural Circuits*, 8:146, 2014. ISSN 1662-5110 (Electronic) 1662-5110 (Linking). doi: 10.3389/fncir.2014.00146.

Tom J. Wills, Francesca Cacucci, Neil Burgess, and John O'Keefe. Development of the hippocampal cognitive map in preweaning rats. *Science*, 328(5985):1573–1576, 2010. doi: 10.1126/science.1188224.

Thomas Wolbers, Cornelius Weiller, and Christian Büchel. Neural foundations of emerging route knowledge in complex spatial environments. *Cognitive Brain Research*, 21(3):401–411, 2004. ISSN 0926-6410. doi: https://doi.org/10.1016/j.cogbrainres.2004.06.013.

Jinwei Xing. Rlcodebase: Pytorch codebase for deep reinforcement learning algorithms. `https://github.com/KarlXing/RLCodebase`, 2020.

Jinwei Xing, Xinyun Zou, and Jeffrey L Krichmar. Neuromodulated patience for robot and self-driving vehicle navigation. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

Jinwei Xing, Takashi Nagata, Kexin Chen, Xinyun Zou, Emre Neftci, and Jeffrey L Krichmar. Domain adaptation in reinforcement learning via latent unified state representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10452–10459, 2021.

Jinwei Xing, Elizabeth R Chrastil, Douglas A Nitz, and Jeffrey L Krichmar. Linking global top-down views to first-person views in the brain. *Proceedings of the National Academy of Sciences*, 119(45):e2202024119, 2022a.

Jinwei Xing, Takashi Nagata, Xinyun Zou, Emre Neftci, and Jeffrey L Krichmar. Policy distillation with selective input gradient regularization for efficient interpretability. *arXiv preprint arXiv:2205.08685*, 2022b.

Jinwei Xing, Xinyun Zou, Praveen K Pilly, Nicholas A Ketz, and Jeffrey L Krichmar. Adapting to environment changes through neuromodulation of reinforcement learning. In *From Animals to Animats 16: 16th International Conference on Simulation of Adaptive Behavior, SAB 2022, Cergy-Pontoise, France, September 20–23, 2022, Proceedings*, pages 115–126. Springer, 2022c.

Jinwei Xing, Takashi Nagata, Xinyun Zou, Emre Neftci, and Jeffrey L Krichmar. Achieving efficient interpretability of reinforcement learning via policy distillation and selective input gradient regularization. *Neural Networks*, 161:228–241, 2023.

D. L. Yamins and J. J. DiCarlo. Eight open questions in the computational modeling of higher sensory cortex. *Curr Opin Neurobiol*, 37:114–120, 2016. ISSN 1873-6882 (Electronic) 0959-4388 (Linking). doi: 10.1016/j.conb.2016.02.001.

Angela J Yu and Peter Dayan. Uncertainty, neuromodulation, and attention. *Neuron*, 46 (4):681–692, 2005.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf`.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21024–21037. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/f0eb6568ea114ba6e293f903c34d7488-Paper.pdf`.

Hui Zhang, Milagros Copara, and Arne D. Ekstrom. Differential recruitment of brain networks following route and cartographic map learning of spatial environments. *PLOS ONE*, 7(9):e44886, 2012. doi: 10.1371/journal.pone.0044886.

Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

Li Zhang, Xin Li, Mingzhong Wang, and Andong Tian. Off-policy differentiable logic reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 617–632. Springer, 2021.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016.

Huihui Zhou and Robert Desimone. Feature-based attention in the frontal eye field and area v4 during visual search. *Neuron*, 70(6):1205–1217, 2011.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Xinyun Zou, Soheil Kolouri, Praveen K Pilly, and Jeffrey L Krichmar. Neuromodulated attention and goal-driven perception in uncertain domains. *Neural Networks*, 125:56–69, 2020.