Active physics learning - World modelling with intrinsically motivated RL

Mingxuan Mei

Master of Science
Data Science
School of Informatics
University of Edinburgh
2025

Abstract

Predicting the movement of unfamiliar physical objects is a complex problem especially when unobservable physical properties like different masses with a same appearance and magnetic forces between objects are involved. But human can often perform interrogatory interactions with objects to reveal those motion patterns, and update predictions according to observed trajectories. These curious actions are often instinctive, their mechanism and motivations still remain mysterious.

This work studied the mechanism behind those actions when human actively acquire knowledge from physical environments. We implemented an dual-model active learning system with reinforcement learning to actively learn and predict objects' motion in a 2D simulated world involving unobservable physical properties like mass and magnetic forces. We adopted intrinsic reward signals measuring predictions' expected improvements to guide a reinforcement learning agent to actively reveal objects' motion patterns when encountering a new world with unseen physical properties. The trained agent is able to accelerate the motion prediction's supervised learning progress.

Acknowledgements

I would like to give my thanks to my supervisor Neil Bramley for his help and inspiration during this dissertation, and also my colleague Héctor Otero Mediero for his excellent idea about the action space we designed for our RL agent.

Table of Contents

Chapter 1

Introduction

Predicting objects' future movement in a short period of time is an instinctive ability of human. Even a child who does not have relevant physics knowledge behind these motions can run after a moving football and eventually catch it. If in a complex environment that objects have hidden properties (magnetic forces or different masses with same appearance) that are not directly observable, one may not be able to make accurate predictions at first, but after applying active interactions to these objects, one can still reveal them through specific observations and adjust predictions according to inferred hidden properties. The strategies of these active interactions perform as key method when human acquiring knowledge from environment.

To model such behaviors and study what drives them, we trained an artificial agent using reinforcement learning. Reinforcement learning [7, 6] is a machine learning technique that allows an agent to explore by itself and guides it with reward signals, aims to learn a control policy that can maximise its expected reward. The reward signals can be used to model the motivations behind the agent's learnt behavior. However, in a natural learning context as described above, it is hard to design a extrinsic reward signal that can guide the agent to improve its motion prediction accuracy.

Actually, human's active interactions during active learning are mostly driven by intrinsic motivations like curiosity, enjoyment and interest. Comparing with extrinsic motivations, intrinsic ones usually result in better learning outcomes [13, 30]. Such motivations guide a subject to acquire necessary knowledge and skills to solve tasks only in a future state, while extrinsic motivations drives subject to deal with the task directly [1].

From machine learning perspective, supervised learning [22] can be considered as a extrinsically motivated behavior that takes pairs of inputs and targets, motivated to

produce outputs as close to targets as possible. Although reinforcement learning also have a reward channel that takes external inputs, Barto [3] argued that, its learning framework is particularly suitable for analogs of intrinsically motivated learning.

This research studies intrinsically motivated learning mechanism during a physics motion prediction task within a 2D simulated domain. We designed an active learning system with intrinsically motivated reinforcement learning, which can learn to explore the world and improve it's prediction accuracy through active interactions with environment. The effects of intrinsically motivated learning behavior to a direct supervised learning problem were analyzed. The designed active learning system[1] followed a multi-model concept originate from cognitive science: Intrinsically motivated learning systems [2]. The system's job of active interacting and predicting were assigned to two different models which are called "*Active learning agent*" and "*World predictor*". The "*Active learning agent*" was pre-trained to be able to catch objects in moving, while the "*World predictor*" was pre-trained to have a general knowledge of physical motions, this is because we wanted to model a subject that already has a familiarity with physical environments and study its performance when encounter a new world. Then the two models were trained together in a new world with unseen settings of hidden properties, reward it for how much it is able to improve its ability predict the physics of that new world, to see how fast the system is able to improve its predictions.

This dissertation will first introduce the background of this project, the specific task we wanted to solve and the objectives we aimed to achieve. After that, it will explain the methodologies used in the "*World predictor*", the "*Active learning agent*" and the overall learning system. Then we will present the experiments we conducted and their results, followed by discussion and conclusions.

1.1 Background

Many researches have been done to study the mechanism behind human knowledge acquisition from environment, from [17] to more recent [29, 5], they aimed at understanding how human acquire knowledge through active interaction with the world.

Statistical methods including machine learning have been used to model such active learning behaviors in psychology [28]. Guez et al. [12] adopted bayesian model-based reinforcement learning to maximising extrinsic reward signals, but they used a explicit approach that requires massive knowledge of the world and needs the agent to plan

[1] open source at `https://github.com/mingxuanM/physics_learning_rl`

whole trajectories among possible futures [20], this is unpractical when dealing with complex long-term tasks.

This work follows researches done by Li et al. [15] and Bramley et al. [8], they studied the cognitive mechanisms involved in inferring hidden physical properties in a 2D simulated domain. The 2D simulated domain is a top view of a bounded world with 4 "pucks" of different masses, labeled from "p1" to "p4", similar to our environment shown in Figure 1.1. They have pair-wise forces of attraction and repulsion like magnetic force between each other, a puck can not go over another, which means they will collide when goes together. Bramley et al. [8] studied human action patterns when actively inferring latent properties (relative masses and pair-wise forces) between objects in this environment. The subjects interact with the environment by controlling a mouse cursor to perform click and hold on a puck to drag it, only one puck can be controlled at a time. Meanwhile, subjects need to observe motions of pucks, and infer the relative mass or pair-wise forces across a range of trials with different ground truth of the latent properties and different starting states. In their second experiment, Bramley et al. [8] had 40 subjects to complete this active inferring task, each with 20 trials. They adopted a simulation-based inference model to evaluate the physical properties generated by subjects, and categorised different action sequences into meaningful strategies then compared the number of observed instance of each strategy during trials with two different goals (inferring relative masses or inferring pair-wise forces). They found that the participants tended to do more *encroaching* (dragging a puck closer to another) when inferring forces while more *launch* (grabbing a puck and throwing it towards another) and *throw* (similar to *launch*, just not aiming at any pucks) when inferring masses.

Building on [8], Li et al. [15] designed and implemented a reinforcement learning agent to conduct a similar task. The agent was trained to minimise its uncertainty about physical properties in a similar 2D environment, intrinsic reward signal was implemented to measure the improvement on properties estimation. They adopted a simulation based inference and Bayesian estimation method to update the agent's belief of hidden physical properties. The belief of hidden physical properties was updated as a posterior, with prior as a uniform distribution over all possible combinations of hidden properties, the likelihood was calculated by comparing the observed trajectories against simulated trajectories with each one of the possible property combinations. The reward was given by how much the agent's uncertainty (entropy) about hidden properties was decreased after one action. This reward calculation process requires an

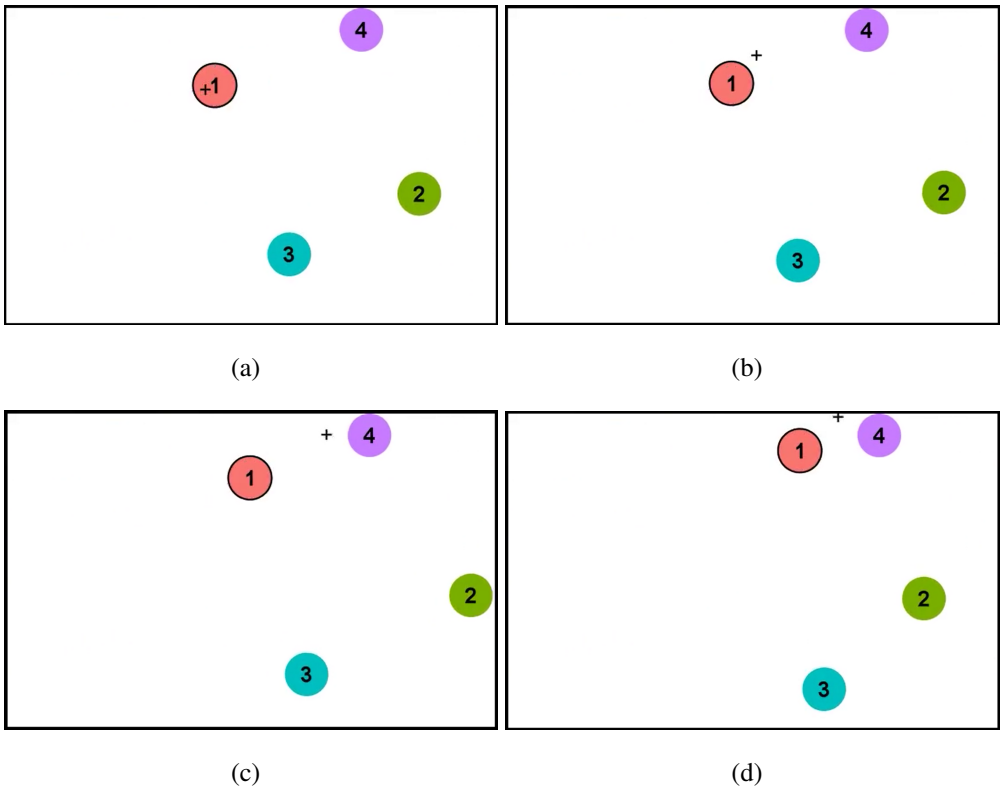

Figure 1.1: A sequence of frames showing the agent catch and drag object 1 using the mouse cursor (the black cross)

external simulator to simulate the estimated trajectories for all of the possible property combinations[2], which is extremely time consuming and not practical as such simulator do not exist in human active learning context.

Li et al. [15] analyzed the agent's performance and potential patterns in the learnt control policies and succeed in showing hints of human-like action sequence to uncover the latent physics properties. They also found similar goal-specific (inferring mass or force) action patterns as Bramley et al. [8] found in human experiments. However, they did not study the meaning and relationship of their reward function with real human learning habits and they did not analyzed the effect of using intrinsic motivation in a RL agent to help a original supervised learning problem.

There are also some problems with the action space and interaction rule of the RL agent designed by Li et al. [15]. They implemented a high dimensional action space with 645 possible actions at each state, yet still had strict constrains on the agent's behavior. They used a set of interpolation functions as actual trajectory of the mouse

[2]Li et al. [15] has 2^{10} distinct mass-force combinations in total

cursor, combined with different directions and controlling objects. As a result, the agent will move the cursor alone the shape of one of the interpolation functions in a selected direction at a constant speed, which is not a natural simulation of human behavior. Also, the interaction rule designed by Li et al. [15] does not require the RL agent to catch any object before controlling it, which is more like using the mouse to attract a puck from distance rather than dragging it.

We would like to study the intrinsically motivated learning mechanism and relate our learning system with it, by designing a more convincing synthesis of human learning system. We are also going to study the effect of intrinsic motivation, compare the results from our intrinsically motivated learning system to a normal supervised learning problem (motion prediction with neural networks).

1.2 Task

In this research, we took a step further and trained a reinforcement learning agent to actively explore the environment and learn to predict trajectories of pucks with hidden properties. We want the agent to be able to interact with pucks in different worlds with unseen hidden properties and improve its motion prediction accuracy in those worlds faster than it could by merely observing them.

This trajectory prediction task is actually a super-set of the previous hidden property inferring task in [8, 15], and it is harder and more general because the moving trajectories over a specific period of time result from pucks' starting states (location and velocity), hidden properties and interactions from the mouse. If the agent can accurately predict pucks' trajectories, it have implicitly inferred those influences including hidden properties.

The same 2D environment as used by Bramley et al. [8] was adopted in this work, whose physical mechanism is slightly different from the 2D environment used by Li et al. [15]. As mentioned above, this simulated environment is a 2D bounded rectangle with size of 6×4 meter, containing 4 circular pucks each with radius of 0.25 meter and different masses, they interact with each other through collisions and pair-wise forces of attraction and repulsion [18]. To limit the number of possible world settings which are defined by combinations of hidden properties, we adopted the same set of possible settings of mass and pair-wise forces used in [8]. The masses are defined with a R^4 vector where each element represents the mass of an object that takes value 1 or 2 kg. In each world setup, only one object can have mass 2 kg while all others

are 1 kg, in this case, the masses setting can also be reflected by a one dimensional number/character indicating which object takes mass 2 while all others are 1. The pair-wise forces are defined with a R^6 matrix, where each element takes value within $(-3, 0, 3)m/s^2$, indicating the forces between two objects[3], positive element represents attractive force, negative element represents repulsive force while zero represents no force. The overall world setup can be defined with a 7 dimensional vector where the first 6 elements are pair-wise forces, the 7th element indicates which object is 2kg[4].

We also refined the interaction rule of the RL agent from [15]. In this work, we wanted to design an agent that has more comparable action sequences with human, thus the agent has to control the mouse cursor to interact with the pucks in a similar manner as human: it can only drag a puck after move the cursor above the target puck and then click the mouse. In this case, we also needed to train the agent to catch a puck with the mouse first.

1.3 Objectives

Our first objective is to design a new action space to replace the one used in [15], the new action space needs to be simpler and more flexible that allows the agent to perform more human-like action sequences.

The main objective is to design and implement an active learning agent capable of completing the tasks described above. The agent is designed to model a subject that already has a familiarity with physical environments, it will be trained on a diverse set of both passive and active trials data, optimising its ability to predict upcoming dynamics in held out test worlds. Then it will explore and learn in test worlds with held out properties, we are going to measure how quickly it is able to improve its predictions in those new worlds (i.e. how quickly it can fine tune its predictions to account for the specific masses and forces in this world). The agent's actions will be reinforced by its improvements on predictions.

The whole training process and final trained agent will be compared with a single world predictor trained without agent interactions or with interactions from a random act agent. By doing that, we aimed to prove that the world predictor from this active learning system converges faster and performs better than the other two.

We referenced a concept called intrinsically motivated learning systems [2] and

[3] $[f_{ab}, f_{ac}, f_{ad}, f_{bc}, f_{bd}, f_{cd}]$

[4] e.g. [0,3,0,3,-3,0,B]

divided the active interactions and world prediction into two models. The next chapter will introduce the intrinsically motivated learning systems and explain its relationship to our active learning system.

Chapter 2

Method - Intrinsically motivated learning systems

This chapter will introduce the overall structure of the learning system and the concept that inspired this design.

Intrinsically motivated learning systems is a long existed concept, inspired by human's intrinsically motivated learning mechanism from human psychology [9, 10], describes a system that can learn by intrinsic motivation (curiosity, interest) for potential tasks in a later stage. They are firstly introduced by [3, 19, 23] to develop robots that can autonomously acquire different knowledge and skills. The trained system could learns to master necessary skills to those tasks more efficiently and faster than learning directly from scratch [2].

To design such a system actually do not required any nova architecture, the abstract nature of RL can work with analogs of both intrinsic and extrinsic reward signals [3]. Schmidhuber [26] supported this idea and stated that the same goal-directed RL can be used to implement curiosity and boredom driven agent. The main problem is "dening specic mechanisms for generating reward signals" [3].

Following this idea, Schmidhuber [27] presented a similar system for active world exploration and learning, which is perfectly suitable to the tasks of this work, he concluded that such a system consists of 4 ingredients below:

1. A world model, essentially a predictor that learn to predict world states, reflecting current knowledge to the world.
2. A learning algorithm that continually improves the world model.
3. Intrinsic reward signals that measures how much the world model is improved.
4. A reinforcement learner, who receives these reward signals and learns a control

policy to maximise its expected reward.

Schmidhuber [27] also suggested some possible ways for generating intrinsic reward signals for measuring the improvements of prediction, among which two ways relative to prediction error were selected as our potential reward design:

(a). Reward signal proportional to predictor's prediction error [26]

(b). Reward signal proportional to to expected improvements of prediction error [24, 25].

Where the improvement is reflected by the first-derivative of prediction error during training, calculated by the expected reduction of error after one training step.

Our active learning system followed this 4-ingredient concept and adopted a dual-model structure whose components are called "*World predictor*" and "*Active learning agent*". Gradient descent [4] will be the learning algorithm that improves the "*World predictor*", and the reward signals *(a)* or *(b)* will be the intrinsic reward signals to guide the "*Active learning agent*". During training, the "*Active learning agent*" receives reward signals and takes actions based on current world state to influence the environment, while the "*World predictor*" is trained to predict future world state given previous states and actions from the agent.

The following two chapters will introduce the design details and training methods of each part of the active learning system.

Chapter 3

Method - World predictor

World predictor is the neural network represents the knowledge of the learning system to the world, it needs to learn to predict the the pucks' movements.

3.1 Data

The training data of world predictor contains 800 human experiment trials data collected by experiment 2 of [8], and 200 generated trials from the 2D environment simulator. 70% of generated trials are passive (no mouse influences) and 30% of generated trials has no pair-wise forces between pucks (these two properties are set independently, randomly distributed in generated data set). The simulator is developed with the JavaScript implementation of box2d physics simulator engine. The human experiment data is anonymous and public accessible from openICPSR website [1].

Test data of world predictor is also generated by the simulator, with random hidden properties and starting state (initial locations and velocity), half of the test cases are passive trials without mouse influence, another half have random sampled mouse control trajectories. The starting state (locations & velocities) in both generated training and test trials are uniformly sampled within constrains: pucks' initial locations are within the world boundary $x \in (0,6); y \in (0,4)$ $meter$ and no overlap with each other; initial velocities are within $v_x \in (-10,10); v_y \in (-10,10)$ $meter/second$.

These data are the records of frames of trials, each trial is 1800 frames long. One frame represents 1/60 second, contains x, y coordinates and x, y velocity of all 4 pucks as well as x, y coordinates of mouse cursor and a 4-bit one-hot encoded representation

[1] https://www.openicpsr.org/openicpsr/project/100799/version/V1/view?path=/openicpsr/100799/\fcr:versions/V1

of which puck is currently under control.

The data format of one frame (22 dimension), where $p1$ to $p4$ means the objects 1 to 4:

$$[\, X_{mouse}, Y_{mouse}, 0, 0, 0, 0, \quad X_{p1}, Y_{p1}, Vx_{p1}, Vy_{p1}, \quad X_{p2}, Y_{p2}, Vx_{p2}, Vy_{p2},$$
$$X_{p3}, Y_{p3}, Vx_{p3}, Vy_{p3}, \quad X_{p4}, Y_{p4}, Vx_{p4}, Vy_{p4} \,]$$

3.2 Network structure

World predictor's job can be simplified as a time-series forecasting task: taking inputs as data from a few previous time-steps to predict the next frame. As mentioned in last section, each frame in input data contains 22 elements[2]. Aside from these information, there are still more motion properties like acceleration and rotation (changes in moving direction) that can only be inferred from a sequence of frames. RNN [21] and LSTM [14] will be the most appropriate network structure for this task. Actually there is previous research using RNNs to simulate physical environments [32, 16, 11].

The length of input sequence determines how many frames the model can take to make the prediction. Considering efficiency, longer input length will need more computation and longer training and evaluation time, but too short input length can not provide enough information to make accurate predictions. In this case, 5 frames was chosen as the input length to world predictor, because 5 frames can cover enough trajectories over a possible collision between pucks, and is enough to calculate their acceleration.

The output as prediction of pucks motions will only contain 16 elements [3]. Thus, the first layer of the world predictor is a LSTM layer with input dimension of 5×22 and output dimension of 22 as well, followed by a tanh activation layer. Building on that, one Dense layer is added to calculate and transform the 22 dimensional middle output to 16 dimensional final output. No activation is added to the Dense layer, as velocity in the final output is not bonded.

The common usage of LSTM on time-series predictions is to predict the next time-step directly, but Ehrhardt et al. [11] stated that predicting the *changes* from the last input time-step to the target time-step will improve prediction accuracy. The final prediction of the target time-step can be calculated by adding the predicted changes to the last input time-step, hence we use *Additive model* to refer to this model, and *Direct*

[2] world predictor input dimension: $4 \times 4 + 2 + 4 = 22$

[3] world predictor output dimension: [x, y, vx, vy] of all 4 pucks, $4 \times 4 = 16$

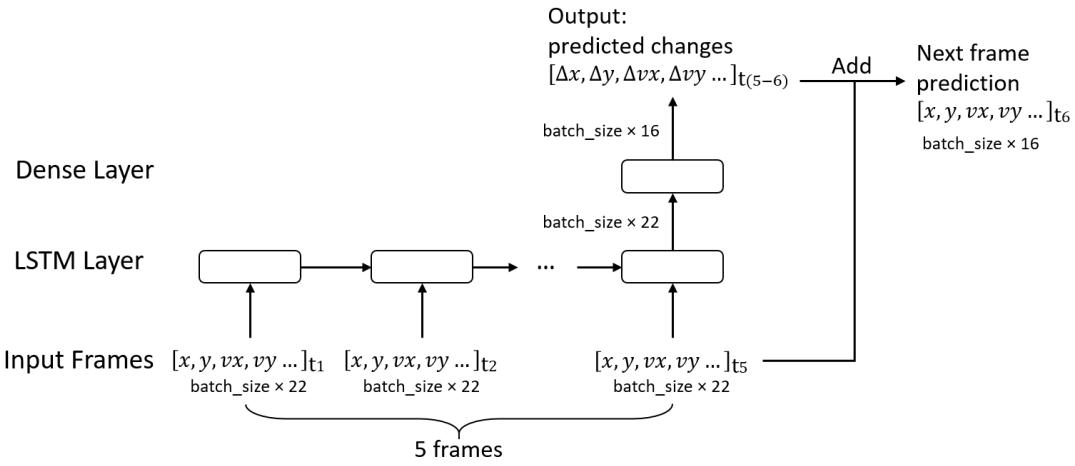

Figure 3.1: World predictor network structure, *Additive model*; The final prediction of the target time-step is calculated by adding the predicted changes to the last input time-step

model to refer to the normal model that predict next time-step directly. Experiment was conducted to assess both methods in Section 6.2, the results proved that *Additive model* outperformed *Direct model*, thus the final world predictor network was designed in this way (Figure 3.1).

3.3 Pre-training

Before being trained within the active learning system, the world predictor needs to be trained on general training set first, this is because we want our system begin as a subject that already has a familiarity with physical environments. In order to make the predictor have a general and brief knowledge of physical motions, training data set contains trials with a diverse set of world settings, and both passive and active trials.

Table 3.1: Motion elements' weights in the average loss and mean standard deviations

motion element	calculation	mean std	weight
x	$(\text{std}(x_{p1}) + \text{std}(x_{p2}) + \text{std}(x_{p3}) + \text{std}(x_{p4}))/4$	1.6973	0.2676
y	$(\text{std}(y_{p1}) + \text{std}(y_{p2}) + \text{std}(y_{p3}) + \text{std}(y_{p4}))/4$	1.0517	0.1658
vx	$(\text{std}(vx_{p1}) + \text{std}(vx_{p2}) + \text{std}(vx_{p3}) + \text{std}(vx_{p4}))/4$	1.7830	0.2811
vy	$(\text{std}(vy_{p1}) + \text{std}(vy_{p2}) + \text{std}(vy_{p3}) + \text{std}(vy_{p4}))/4$	1.8112	0.2855
sum		6.3432	1

During training, a direct L2 loss is the squared difference between model prediction

and target frame for each element, which is 16-dimension. A mean over 4 pucks for each motion element (x,y,vx,vy) are taken to get a 4-dimension loss. Then a weighted average is taken over the 4 elements to get the final loss for training and testing. The weight of each element is calculated by this element's standard deviation (average over four pucks) in the general training set over the sum of four elements' standard deviations. The mean standard deviations and the loss weights for each element is given by Table 3.1. Where x_{p1} means the x coordinate of puck 1 and so on.

The pre-trained world predictor in the active learning system was trained on the training set for 20 epochs with a relatively low learning rate of 1e-5. Training details can be seen in Section 6.2.

Chapter 4

Method - RQN learning agent

Model-free reinforcement learning requires the RL agent to learn an optimal control policy to maximise the cumulative reward defined in a Markov Decision Processes (MDP). The MDP can be given in the form of *(S, A, R, S', A')* where *S* is current state; *A* is the current action taken at state *S*; *R* is the reward caused by *A* at *S*; *S'* is the next state caused by *A* at *S*; and *A'* is the next action taken at *S'*.

The follow sections will introduce the state, actions and reward signals defined for this active learning agent, and then introduce the learning algorithm and model used to learn the control policy.

4.1 State representation

In this problem, the state is same as inputs to the world predictor network: pucks' x, y location and x, y velocity plus mouse's x, y location and controlling object (22 elements). Also, in order to give enough information to the agent to infer the pucks' latent proprieties and other motion factors, a sequence of last 5 frames from time-step t-4, t-3, t-2, t-1 to time-step t is used as the state at time t for the MDP.

4.2 Action space

As mentioned in Section 1.3, we need to design a simple (low dimensional) and flexible action space for the agent to perform natural human-like action sequences.

We considered the mouse cursor as an flouting object moving above the pucks in the 2D world, whose motion is represented as x, y coordinates and x, y velocity. With

no force, the cursor will be decelerated by a scale of γ_v *(velocity decay rate)* in each frame until fully stop[1].

The possible actions for the agent is to *push (accelerate)* the cursor in one of four directions[2] or try to *click* mouse or *no action* at all, 6 possible actions in total. The push will add a scalar *a* `acceleration` to velocity in the chosen direction in each frame[3].

Each action will last for 5 frames, this length is very short compared with 40 frames in [15], this is because short actions can be combined in any way in a sequence to form more possible mouse trajectories, this is left to the agent for itself to learn.

Action influence to cursor velocity in each frame:

- NO ACTION: $V_x = V_x \times \gamma_v; \quad V_y = V_y \times \gamma_v$
- ACCELERATE IN +X: $V_x = V_x + a$
- ACCELERATE IN -X: $V_x = V_x - a$
- ACCELERATE IN +Y: $V_y = V_y + a$
- ACCELERATE IN -Y: $V_y = V_y - a$

When the agent selects *click* action, it clicks once at the first frame of this action, at that time, if the cursor is above any puck (distance from cursor to a puck's center is less than 0.25 meter), this puck will be taken under control until the agent choose *click* again. If the agent fails to catch any thing, this action will be same as *no action*. In this way, the agent will be able to perform a continued dragging action sequence.

4.3 Reward signal

This section will first explain the reward signal for catching pucks and two different intrinsic reward signal designs for active world learning. Two parts of reward signals will both be given when the agent is trained within the active learning system.

4.3.1 Reward for catching pucks

To catch an object, the agent has to control the mouse cursor to approach the target first then take *click* action to catch it. In this case, we designed two parts of reward signal for this goal: approaching reward and catching reward.

[1] γ_v *(velocity decay rate)* is set to 0.9 per frame

[2] four directions: +x, -x, +y, -y

[3] *a* `acceleration` is set to 1 m/s per frame

The reward for a successful catching is set to a constant number that is larger than the maximum value of approaching reward, so that the agent's policy will not be dominated by approaching rather than catching pucks.

In order to let the agent catch a puck as soon as possible, we reward it with the reduced distance to the puck that is closet to the mouse cursor at beginning of an action. To simplify the reward calculation, x and y distance are tracked separately. At the first frame of each action, the distance from cursor to the nearest puck is recorded as x_0 and y_0; at the end of each action, the distance to the same object is recorded as x_t and y_t. The approaching reward for a single action is set to be bounded within the range from 0 to 2. But in this way, the reward for reduced x distance is given by $x_0 - x_t$, which can not get larger than x_0, a potential problem could happen when the cursor is very close to the nearest puck, the approaching reward will get small on the contrary. To solve this, we divide the reduced distance by the beginning distance. The final reward signals for one action are listed below:

- If catching a puck: 5
- If not, approaching the nearest puck: $max(\frac{x_0-x_t}{x_0}, 0) + max(\frac{y_0-y_t}{y_0}, 0)$

4.3.2 Intrinsic reward for active world learning

To help world predictor learning in our active learning system as discussed in Chapter 2, the agent can have intrinsic reward in two different ways: reward proportional to predictor loss [26], or proportional to first-derivative of the predictor loss w.r.t training steps [25, 24]. This first-derivative is given by the change of a loss before and after a training step.

Giving reward proportional to predictor loss is a simple way to guide the agent to explore those trajectories that the predictor is not good at. The problems of this reward signal is that it may lead the agent to hover in random noises where the predictor may never be able to learn anything, but this problem do not exist in this learning environment as all the pucks is moving according to Newtonian mechanics.

Reward proportional to first derivative of the predictor loss during training solved the problem of trapping agent in noises. But this reward has to be calculated when both active agent and world predictor are being trained simultaneously, during which the predictor network's loss is changing, which may cause the agent learning unstable. More details will be discussed in Chapter 5.2

During training, each frame is incorporated as a training data point for the world predictor network, given inputs as previous 5 frames. So that each action of the agent that lasts 5 frames will give the predictor 5 training steps, and the resulting intrinsic reward signals are given in the forms below:

- proportional to loss:

$$\lambda_r \times \frac{1}{L_{action}} \sum_{t=T}^{T+L_{action}} loss_t$$

- proportional to first derivative of loss:

$$\lambda_r \times \frac{1}{L_{action}} \sum_{t=T}^{T+L_{action}} (loss_t - loss_{t-1})$$

Where λ_r (*reward rate*) is the ratio of reward over mean loss or mean first derivative of loss, set to 1 when the agent is not controlling anything and 1.5 when one of the pucks is under control;

L_{action} is the length of an action, set to 5 frames;

$loss_{t-1}$ is the world prediction loss before predictor's training step t;

$loss_t$ is the world prediction loss after predictor's training step t, on the same training point;

Both reward signals were implemented and experiments were conducted to compare their effect to the training process.

4.4 Network structure

The reinforcement learning agent adopts Q-learning [31] as the learning algorithm. Q-learning is a model-free off-policy reinforcement learning algorithm that update Q-value of state action pair (s,a) as:

Update Target: $r + \gamma \max_{a'} Q(s', a')$

TD loss: $Q(s,a) - (r + \gamma \max_{a'} Q(s', a'))$

Updated Q-value: $\hat{Q} = Q(s,a) - \alpha(Q(s,a) - (r + \gamma \max_{a'} Q(s', a'))$

Where α is the learning rate; γ is the reward discount rate, set to 0.9.

Q-value at (s,a) represents the expected cumulative discounted reward for action a at state s. In conventional practice, Q-values are stored and updated in a tabular form, with each (s,a) pair as an entry. However, as Li et al. [15] stated, in this problem the locations and velocity of objects all continuous throughout the world, thus a Deep Q-Network (DQN) was adopted to approximate the Q-values.

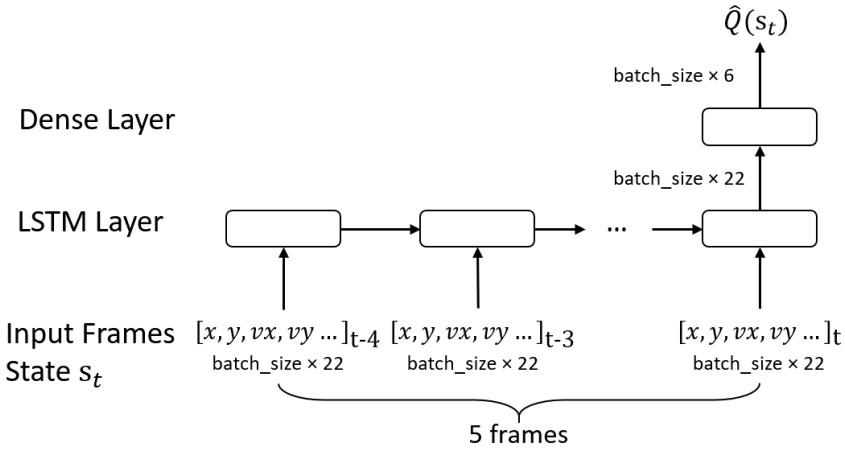

Figure 4.1: RQN learning network structure

Li et al. [15] also suggested a recurrent neural network solution called RQN, in which they used a LSTM layer and a Dense layer as the DQN. This is for the same reason as the network design in world predictor: the input state is a sequence of previous frames, LSTM can learn the relationship through the time-series. Their experiments proved that this RQN solution works better than a normal Multi-Layer Perceptron structure. Thus, we took the same network design, its structure is shown in Figure 4.1.

Because this is a bootstrap learning algorithm, update target involves network output $Q(s', a')$ which is unstable during training. In order to improve the learning stability, we used a target network and update the learning recurrent Q-network (RQN) using semi-gradient descent same as [15].

4.5 Pre-training

Similar to the world predictor, Before being trained within the active learning system, the RL agent needs to be individually trained to catch pucks first. For the same reason as mentioned in Section 3.3, a child should have the ability to catch a moving object despite he/she may not completely understands the world.

The pre-trained RL agent in the active learning system was trained with 10000 simulated episodes and can succeed in catching a puck in 92% of 200 test episodes with average number of actions of 78.49 (6.54 seconds). Training details can be seen in Section 6.3.

Chapter 5

Method - System training frameworks

This chapter will introduce the active learning system as a whole picture. By using two different intrinsic reward signals as mentioned in subSection 4.3.2, the whole learning system will also be different.

By using the reward signal proportional to world predictor's loss, as only calculating loss does not requires the predictor be trained at the same time, these two networks can be trained separately. A benefit of this is that the predictor's loss during the training of RL agent will stay constant, which ensures the stability of RL training, two networks can both be trained to converge during each one's training.

The reward signal proportional to first derivative of world predictor's loss w.r.t training steps, on the other hand, requires both networks be trained simultaneously. During training, each training step will change the loss as well as the loss drop of world predictor network. As a consequence, the reward signal to the RL agent for a same [state, action] pair will change, this will lead to instability of RL training. One possible solution is to use a relatively low learning rate for the world predictor and a relatively high learning rate for the RL agent, in order to limit the change of predictor's loss and accelerate the training of RL agent.

The details of both training frameworks will be explained in the following two sections.

5.1 *Separate Framework*: Separate training with loss reward signal

With both pre-trained RL agent and pre-trained world predictor ready, we choose a world setup of hidden properties to start the active learning system, this world setup will not change during the whole system training.

The training framework begins with the training of RL agent, learning the losses of current world predictor. After the RL agent converges to the losses of current world predictor, it will be used to generate a new training set within the simulated world of the same hidden properties. Then, the world predictor will be trained on the new training set. After it converges to the current optimal, we go back to the first step, and let the RL agent to learn the predictor's new losses and so on. This learning loop can be illustrated as Figure 5.1.

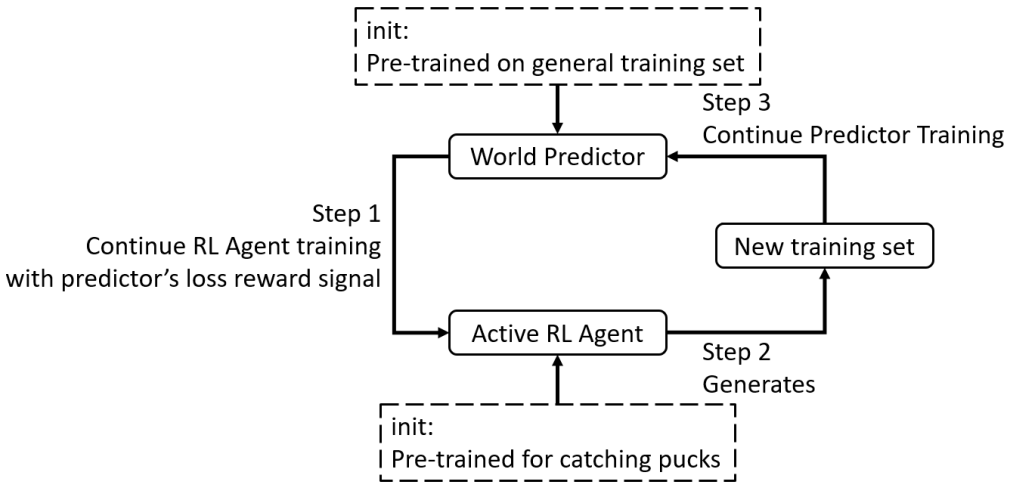

Figure 5.1: Training loop of learning framework 1 with loss reward signal. Active RL Agent and World Predictor are trained separately

5.2 *Concurrent Framework*: Concurrent training with loss's first-derivative reward signal

In this framework, the pre-trained active RL agent and pre-trained world predictor are trained simultaneously. At time t with state s_t, a training step begins with an action a_t sampled from training policy (epsilon greedy policy), then the environment returns next 5 frames as next state s_{t+1} and extrinsic reward $extrinsic\_r_t$. These 5 frames

will be used as a new set of training data for the world predictor. After 5 training steps, the mean loss drop over the 5 steps will be given to the agent as intrinsic reward $intrinsic\_r_t$. Both extrinsic and intrinsic rewards are added to give r_t for the RL agent to run one training step. Then, sample a_{t+1} at s_{t+1} to continue training. This learning steps can be illustrated as Figure 5.2.

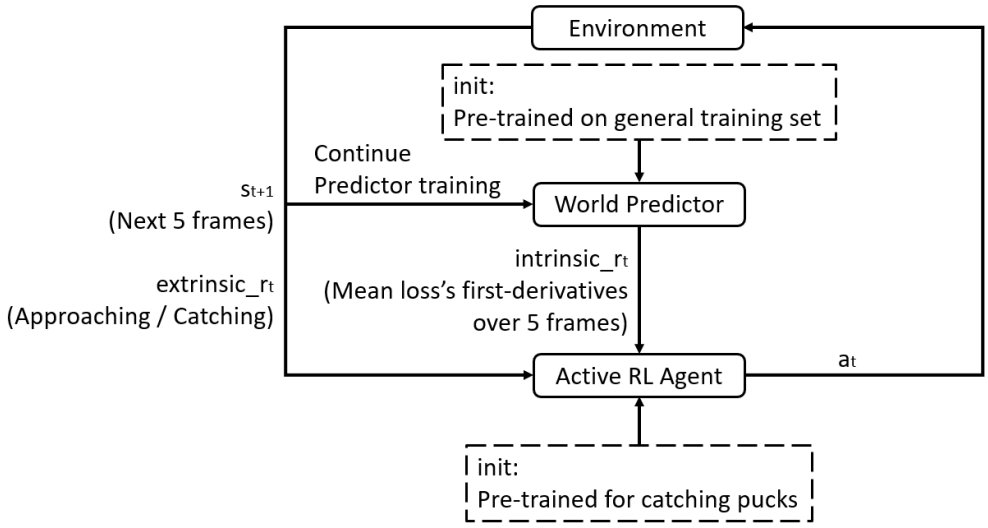

Figure 5.2: Training step of learning framework 2 with loss's first-derivative reward signal. Active RL Agent and World Predictor are trained simultaneously.

Chapter 6

Experiments

In this chapter, we present some training details of different parts of our active learning system and the evaluation about their performance. Different evaluation metrics were setup for each individual part, and the final learnt control policy will be examined to see if there are any potential patterns in the generated action sequences.

6.1 Device setup

All model training and testing involved in this chapter was carried out on an external RTX 2070 GPU connected to a XPS 13-9350 laptop.

6.2 World predictor pre-training

The world predictor network was pre-trained on general training set (Section 3.1) to have a general knowledge of the physical motion mechanism. But during network design, we need to train the network thoroughly to make sure that it has the ability to make accurate predictions. Two baselines in this case were setup to evaluate its performance:

- *Random baseline*: Take the frame at the target time-step from a randomly sampled trial sequence from the general training set as prediction.
- *Infinite inertia baseline*: Assume pucks stop moving just before the target time-step, use the last (5_{th}) frame of the input sequence as prediction.

Here we trained two model designs on the general training set and compared their test losses against each other and the baselines. The *Direct model* is to use the network

Table 6.1: Test losses of world predictor training

Model & Baseline	Learning rate	Test losses of models trained with different number of epochs				
		10	20	30	40	50
Additive model	1e-4	0.2755	0.2556	0.2492	0.2508	0.2546
	1e-5	0.3496	0.3425	0.3321	0.3217	0.3127
Direct model	1e-4	0.5759	0.6072	0.6159	0.6176	0.6229
	1e-5	0.6765	0.6250	0.6040	0.5909	0.5835
Random baseline		8.7052				
Infinite inertia baseline		0.3747				

output as the prediction directly. While *Additive model* is to let the network predict the changes from last input frame to the next frame. Both models were trained on the general training set for 50 epochs with learning rate at 1e-4 and 1e-5. Checkpoints were saved at every 10 epochs and tested on the test set, their test losses are listed in Table 6.1 with baselines. The models' training curves as well as baseline loss are plotted in Figure 6.1. The training batch size was set to 20, so that each training sequence (1800 frames long) will have 85 batches[1].

It can be seen from Figure 6.1 that the test loss of *Direct model* can not beat the *Infinite inertia baseline* while *Additive model* trained with both learning rates have lower test losses. Thus the *Additive model* structure was selected as the world predictor.

The pre-trained predictor to be used in the active learning system should at least have a better performance than the baseline but should not be trained too much that over-fit to the training set, as it still needs to be trained in the learning system. For this reason, although the additive model with learning rate 1e-4 has lower losses, its test loss begin to increase after 30 epochs. Finally the predictor trained with 1e-5 learning rate and 20 epochs was chosen as the pre-trained world predictor to be used in the active learning system.

6.3 RL agent, catching puck pre-training

The active RL agent was pre-trained on 10000 episodes with only catching reward, the training episodes were terminated when the agent succeed in catching a puck, and

[1] $1800 \div 20 - 5 = 85$; where 5 is the number of input frames

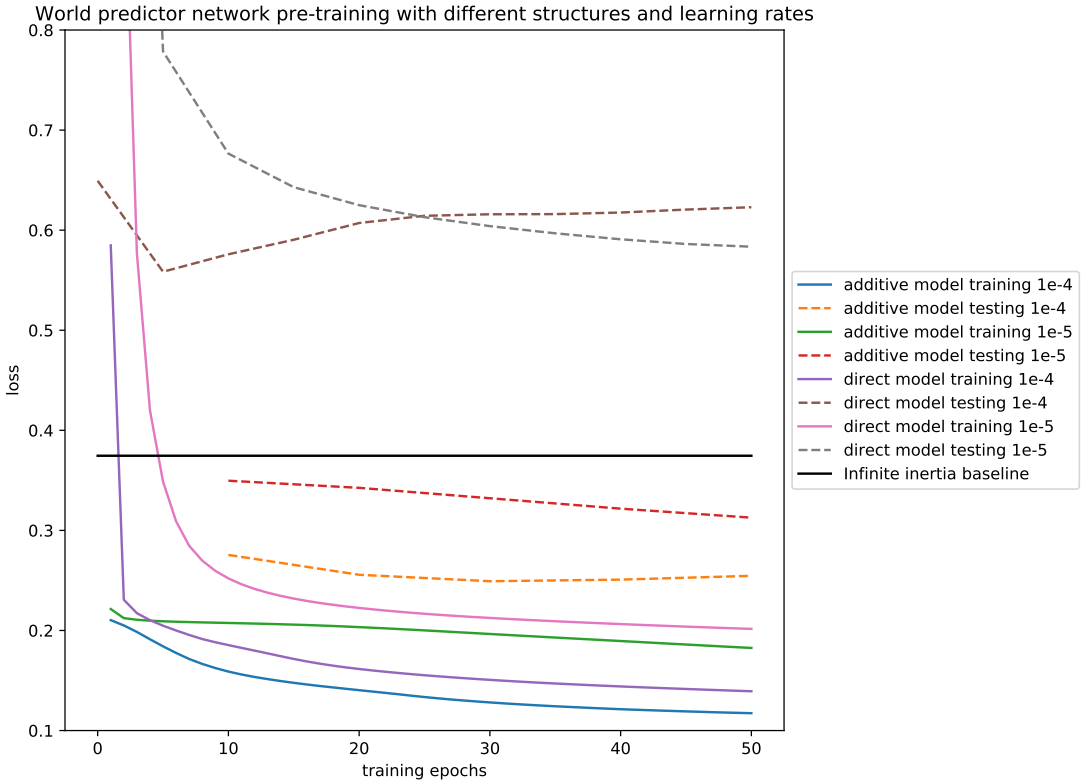

Figure 6.1: World predictor pre-training and testing with different models and learning rates. *Random baseline* test loss is 8.7052, which is too high to plot in this figure.

they had only 500 time-steps maximum to speed up training. Epsilon greedy policy was adopted as the training policy, the epsilon was set to begin with 0.99 and exponentially decay at a rate of 0.9995 after every two training episodes, until it is reduced to 0.01, the minimum epsilon value. Because the approaching and catching reward signals are straight forward to guide the agent to catch its closet object, and the approaching reward avoids any sparse problem during training, the agent does not need to do much exploration. This epsilon decay strategy can assure the first 2730 training episodes have epsilons larger than 0.5 (more tend to explore) and more remaining training episodes can focus on exploit.

Because Q-learning is a bootstrap learning algorithm, it's update target $[r+\gamma \max_{a'} Q(s',a')]$ involves current Q-value estimation $Q(s',a')$ which changes during training, this will leads to unstable training and might make the network unable to converge. In order to improve the learning stability, target network method was used, the parameters in the learning network are copied to the target network after every two training episodes. In this case, the L2 loss becomes:

$$\frac{1}{2}(Q_{learning}(s,a) - (r+\gamma \max_{a'} Q_{target}(s',a')))^2$$

Semi-gradient descent was used to update the learning network's parameters.

After training, the agent was tested on 200 test episodes to calculated the number of successful episodes. The rate of success and average number of actions required to catch a puck were compared with the performance of a random agent, which samples a random action at each time-step.

The mean reward of an action and number of actions required to catch a puck during catch training is presented in Figure 6.2.

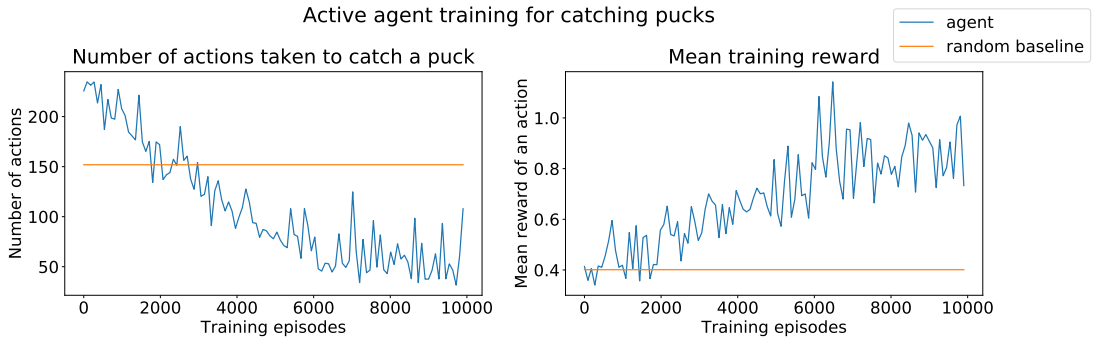

Figure 6.2: Training process of active agent for catching pucks

The success rate and the average number of actions required to catch a puck of the trained agent and random agent is presented in table 6.2. The trained agent is able to catch a puck in 97.5% of test episodes, taken 37.92 actions (189.6 frames, 3.16 seconds) averagely, which is much better and more efficient than a random agent.

Table 6.2: Active agent testing for catching pucks

	success rate (from 200 test episodes)	Average number of actions
Random agent	112/200 (56.0%)	151.88
Pre-trained agent	195/200 (97.5%)	37.92

6.4 Active learning system

After pre-trained the world predictor and the active agent, we made some attempts to train the whole learning system with the training frameworks introduced in Chapter 5.

Due to the limit of time and computational equipment, we can only train the first *Separate framework* (Section 5.1) for one loop; and the second *Concurrent framework* (Section 5.2) for 500 episodes. Because concurrent training is significantly slower

than the separate training, the *Concurrent framework* training is still far from complete, while the *Separate framework* training result can already show some significant superiority over a random baseline agent.

6.4.1 Separate framework training

Two world setups[2] were randomly chosen as the experiment setups, the same process were carried for both setups. The pre-trained active agent was firstly trained on 2000 training episodes with a constant world setup but different pucks' starting states and with intrinsic reward based on the pre-trained predictor's loss. The same epsilon greedy learning policy and target network method as used in Section 6.3 were also adopted here, only the epsilon decay rate was set to 0.995.

Then the trained active agent was set to run 500 test episodes with the same world setup, these 500 trials were recorded as a new active training set for the pre-trained predictor. The new active training set was used to continue train the world predictor with 1e-4 learning rate, 20 batch size, for 50 epochs. After each 5 epochs, the predictor was tested on a generated test set with the same world setup, these test losses during training were plotted in Figure 6.3 for both world setups, along with the test losses of a same pre-trained predictor continue trained on a data set generated by a random agent (pure supervised learning baseline).

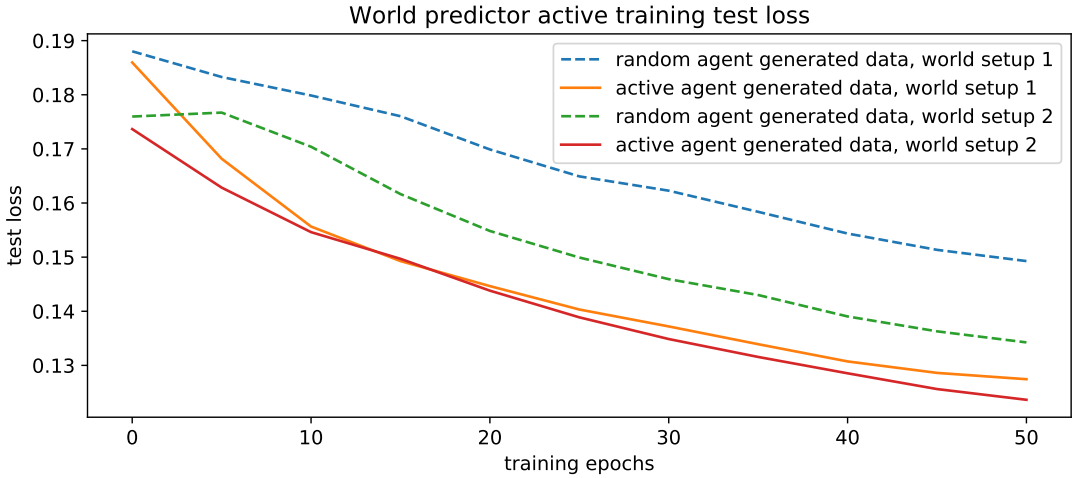

Figure 6.3: Test loss of world predictor continue trained on new training data from active agent and random agent for two different world setups

Figure 6.3 shows the test loss of predictor trained on active-generated data set drops

[2] 1: [0,3,0,3,-3,0,"B"]; 2: [0,0,0,0,0,0,"same"]

faster than the predictor trained on random-generated data set, and the loss drop become more and more slow as trained on more epochs, which indicates that the active-generated data set can accelerate training and help the predictor reaching converge.

6.4.2 concurrent framework training

One of the world setups [0,3,0,3,-3,0,“B”] is used in this experiment, same epsilon greedy learning policy and target network method was also used for active agent training. The 5 frames generated by each action were used to train the world predictor with a learning rate at 1e-5.

Because of the limitation of computational resources, only one environment simulator can run at a time, thus, the training batch size for world predictor can only be one. This limited the training speed, only 500 concurrent training episodes took 10 hours to complete, yet the world predictor was still far from convergence.

The world predictor’s training loss is plotted in Figure 6.4 along with the losses of a same pre-trained world predictor trained with a random agent simultaneously. For the reasons discussed in Section 5, the training is extremely unstable and inefficient, it can hardly see any difference between these two training curves.

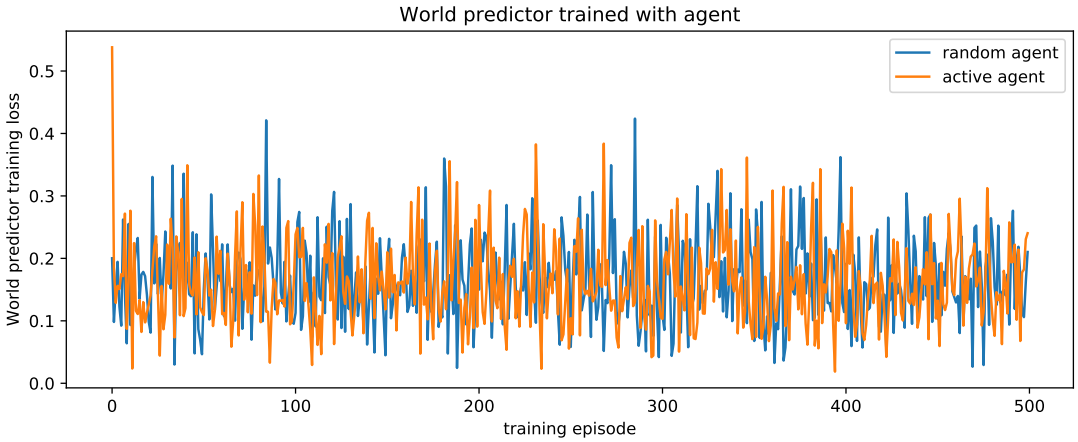

Figure 6.4: Concurrent active training

Chapter 7

Discussion

The result of the experiment in section 6.4.1 proved the active learning system trained with the *Separate framework* is able to improve its motion prediction faster than a random agent. However, because of the limitations on computational equipment and time, this experiment can only be conducted on two test world settings and both for a single training loop, further experiments should be done for more training loops while the number of agent's training episodes and predictor's training epochs can be decreased, to investigate 1: if this improvement is still effective when the predictor has already converged to a local optimal and 2: how much this system can improve its prediction given enough training loops. More test world settings should also be involved in the experiments to generalize these results.

On the other hand, the experiment in section 6.4.2 with *Concurrent framework* did not show any significant difference between the performance of the active learning agent and a random agent. But it is still too early to say that this training framework does not work well as the previous one, as this framework is much more complex, and the time limitation did not allow us to do more experiments to optimise its training process. The value of hyper-parameters play an important role in this framework: the learning rates of two networks control the balance of their training progress; agent's epsilon controls how much should the agent explore rather than exploit; λ_r (*reward rate*) controls the focus of the agent on catching pucks or active learning.

When epsilon is high at beginning, the agent more tends to randomly explore, which is not much different from a random agent. At this stage, predictor's learning rate should be set to a low value to stabilize it's loss change and wait for the agent to learn a useful policy. When agent's epsilon is reduced to a low value, it more tends to exploit, during which the predictor should have a high learning rate to learn the

trajectories influenced by the agent's interactions.

A larger λ_r should be tested in this framework because current active learning reward signal is relatively small. From Figure 6.1, it can be seen that losses are mostly in range between 0.1 and 0.7, which is comparable to the catching pre-training rewards plotted in Figure 6.2 (mostly between 0 and 1). But losses' first derivatives are usually much smaller because the learning rate was set to small values (1e-4 and 1e-5 in this work). To neutralize this effect, λ_r should be set to at least 1e+3.

Finally, similar to the *Separate framework*, this *Concurrent framework* can also have a "big loop" structure by using a fluctuant epsilon: at start, the epsilon is set to decay as normal, but it can be reset and start another decay period when the intrinsic reward becomes small, which means the predictor has converged to a local optimal given current agent's control policy.

Chapter 8

Conclusions

In this dissertation, we implemented an active learning system that can actively interact with an environment to improve its predictions of objects' movements in this environment. This system adopted a dual-model structure, which contains a "*World predictor*" and an "*Active learning agent*". Both models are neural networks with LSTM that takes observed objects' trajectories as a sequence of frames.

The "*Active learning agent*" was trained by Q-learning RL algorithm with deep function approximation and reinforced by intrinsic reward signals that measures how the "*World predictor*" is improved. In order to make the "*Active learning agent*" capable of producing natural and rich human-like action sequences, we implemented a simple yet flexible action space by integrating accelerate action like human's "push" to the mouse cursor and modeling the cursor as a floating object above the environment.

Through experiments with two types of training frameworks (Section 6.4), we found the active learning system trained within the *Separate framework* (reinforce actions with *World predictor's* loss) can improve its prediction significantly faster than a random agent, while the active learning system trained within the *Concurrent framework* (reinforce actions with first derivative of *World predictor's* loss) did not show any superity over a random agent.

The performance of the system trained within the *Separate framework* shows a intrinsically motivated RL agent can learn an efficient control policy to accelerate and improve the convergence of a world-modelling supervised-learning task. This provided a new way to study human's intrinsically motivated learning behavior but more experiments remains to be done to evaluate this system's limitation.

The system trained within the *Concurrent framework* still requires more research to optimize its training process, possible solutions were discussed in Chapter 7.

Bibliography

[1] Baldassarre, G. (2011). What are intrinsic motivations? a biological perspective. In *2011 IEEE international conference on development and learning (ICDL)*, volume 2, pages 1–8. IEEE.

[2] Baldassarre, G. and Mirolli, M. (2013). *Intrinsically motivated learning in natural and artificial systems*. Springer.

[3] Barto, A. G. (2013). Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer.

[4] Barzilai, J. and Borwein, J. M. (1988). Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148.

[5] Battaglia, P. W., Hamrick, J. B., and Tenenbaum, J. B. (2013). Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332.

[6] BERTSEKAS, D. P. (2012). Approximate dynamic programming.

[7] Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA.

[8] Bramley, N. R., Gerstenberg, T., Tenenbaum, J. B., and Gureckis, T. M. (2018). Intuitive experimentation in the physical world. *Cognitive psychology*, 105:9–38.

[9] Deci, E. (1975). Intrinsic motivation plenum. *New York*.

[10] Deci, E. L. and Ryan, R. M. (2010). Intrinsic motivation. *The corsini encyclopedia of psychology*, pages 1–2.

[11] Ehrhardt, S., Monszpart, A., Vedaldi, A., and Mitra, N. (2017). Learning to represent mechanics via long-term extrapolation and interpolation. *arXiv preprint arXiv:1706.02179*.

[12] Guez, A., Silver, D., and Dayan, P. (2012). Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in neural information processing systems*, pages 1025–1033.

[13] Habgood, M. J. and Ainsworth, S. E. (2011). Motivating children to learn effectively: Exploring the value of intrinsic integration in educational games. *The Journal of the Learning Sciences*, 20(2):169–206.

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

[15] Li, S., Sun, Y., Liu, S., Wang, T., Gureckis, T., and Bramley, N. (2019). Active physical inference via reinforcement learning.

[16] Martinez, J., Black, M. J., and Romero, J. (2017). On human motion prediction using recurrent neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[17] McCloskey, M. (1983). Naive theories of motion. *Mental models*, pages 299–324.

[18] Mei, M. (2019). Informatics project proposal: Active physical inference via model-based reinforcement learning.

[19] Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286.

[20] Raiffa, H. and Schlaifer, R. (1961). Applied statistical decision theory.

[21] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[22] Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

[23] Schembri, M., Mirolli, M., and Baldassarre, G. (2007). Evolving internal reinforcers for an intrinsically motivated reinforcement-learning robot. In *2007 IEEE 6th International Conference on Development and Learning*, pages 282–287. IEEE.

[24] Schmidhuber, J. (1991a). Adaptive curiosity and adaptive confidence. Technical report, Technical Report FKI-149-91, Institut f ur Informatik, Technische .

[25] Schmidhuber, J. (1991b). Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE.

[26] Schmidhuber, J. (1991c). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227.

[27] Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

[28] Schulz, E. and Gershman, S. J. (2019). The algorithmic architecture of exploration in the human brain. *Current opinion in neurobiology*, 55:7–14.

[29] Smith, K. and Vul, E. (2014). Looking forwards and backwards: Similarities and differences in prediction and retrodiction. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 36.

[30] Vansteenkiste, M., Simons, J., Lens, W., Sheldon, K. M., and Deci, E. L. (2004). Motivating learning, performance, and persistence: the synergistic effects of intrinsic goal contents and autonomy-supportive contexts. *Journal of personality and social psychology*, 87(2):246.

[31] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[32] Watters, N., Zoran, D., Weber, T., Battaglia, P., Pascanu, R., and Tacchetti, A. (2017). Visual interaction networks: Learning a physics simulator from video. In *Advances in neural information processing systems*, pages 4539–4547.