석사학위논문
Master's Thesis

본질적인 동기 유발 중심의 직관적 물리 학습

Intrinsic Motivation Driven Intuitive Physics Learning using Deep Reinforcement Learning with Intrinsic Reward Normalization

2019

최재원 (崔栽源 Choi, Jae Won)

한국과학기술원

Korea Advanced Institute of Science and Technology

석사학위논문

본질적인 동기 유발 중심의 직관적 물리 학습

2019

최재원

한국과학기술원

전기및전자공학부

본질적인 동기 유발 중심의 직관적 물리 학습

최재원

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2018년 12월 27일

심사위원장 윤성의 (인)

심사위원 이상아 (인)

심사위원 이상완 (인)

Intrinsic Motivation Driven Intuitive Physics Learning using Deep Reinforcement Learning with Intrinsic Reward Normalization

Jae Won Choi

Advisor: sung-eui Yoon

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering

Daejeon, Korea
December 27, 2018

Approved by

sung-eui Yoon
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

MRE
20173597

최재원. 본질적인 동기 유발 중심의 직관적 물리 학습. 전기및전자공학부 . 2019년. 20+iii 쪽. 지도교수: 윤성의. (영문 논문)
Jae Won Choi. Intrinsic Motivation Driven Intuitive Physics Learning using Deep Reinforcement Learning with Intrinsic Reward Normalization. School of Electrical Engineering . 2019. 20+iii pages. Advisor: sung-eui Yoon. (Text in English)

초록

영아들은 주위의 대상을 지속적으로 관찰하고 상호 작용함으로써 실세계 모델을 매우 빠르게 학습하고 구축할 수 있습니다. 영아들이 구축하는 가장 근본적인 직감 중 하나는 직관적 물리입니다. 인간 유아는 추후 학습을 위한 사전 지식으로 사용되는이 모델을 배우고 개발합니다. 인간 유아가 보여준 그러한 행동에 영감을 받아 강화 학습과 통합된 물리 네트워크를 소개합니다. pybullet 3D 물리 엔진을 사용하여 물리 네트워크가 객체의 위치와 속도를 매우 효과적으로 추론 하고, 강화 학습 네트워크는 에이전트가 내재적 동기만을 사용하여 객체와 지속적으로 상호 작용함으로써 모델을 개선 하는 것을 보여주고자 합니다. 또한 직관적 물리 모델을 가장 효과적으로 개선 할 수있는 작업을 효율적으로 선택할 수있는 보상 정규화 트릭을 소개합니다. 우리는 고정 및 비 고정 상태 문제 모두에서 모델을 실험하고 에이전트가 수행하는 다양한 작업의 수와 직관 모델의 정확성을 측정하여, 본 연구의 우수성을 보이고자 합니다.

핵심낱말 직관적 물리, 강화 학습, 내재적 동기

Abstract

At an early age, human infants are able to learn and build a model of the world very quickly by constantly observing and interacting with objects around them. One of the most fundamental intuitions human infants acquire is intuitive physics. Human infants learn and develop these models which later serve as a prior knowledge for further learning. Inspired by such behaviors exhibited by human infants, we introduce a graphical physics network integrated with reinforcement learning. Using pybullet 3D physics engine, we show that our graphical physics network is able to infer object's positions and velocities very effectively and our reinforcement learning network encourages an agent to improve its model by making it continuously interact with objects only using intrinsic motivation. In addition, we introduce a reward normalization trick that allows our agent to efficiently choose actions that can improve its intuitive physics model the most. We experiment our model in both stationary and non-stationary state problems, and measure the number of different actions agent performs and the accuracy of agent's intuition model.

Keywords Cognitive Science, Intuitive Physics, Reinforcement Learning, Intrinsic Motivation

Contents

List of Tables

List of Figures

Chapter 1. Introduction

Various studies in human cognitive science have shown that humans rely extensively on prior knowledge when making decisions. Reinforcement learning agents require hundreds of hours of training to achieve human level performance in ATARI games, but human players only take couple hours to learn and play them at a competent level. This observation begs the question, what prior knowledge do humans have that accelerates their learning process? Recent works [Lake et al.(2017)Lake, Ullman, Tenenbaum, and Gershman] suggest that there are two core ingredients which they call the 'start-up software': intuitive physics and intuitive psychology. These intuition models are present at a very early stage of human development and serve as core knowledge for future learning. But how do human infants build intuition models?

Imagine human infants in a room with toys lying around at a reachable distance. They are constantly grabbing, throwing and performing actions on objects; sometimes they observe the aftermath of their actions, but sometimes they lose interest and move on to a different object. The "child as scientist" view suggests that human infants are intrinsically motivated to conduct their own experiments, discover more information, and eventually learn to distinguish different objects and create richer internal representations of them [Lake et al.(2017)Lake, Ullman, Tenenbaum, and Gershman]. Furthermore, when human infants observe an outcome inconsistent with their expectation, they are often surprised which is apparent from their heightened attention [Baillargeon(2007)]. The mismatch between expectation and actual outcome (also known as expectancy violations) have shown to encourage young children to conduct more experiments and seek more information [**?**, **?**].

Inspired by such behaviors, in this paper, we explore intrinsically motivated intuition model learning that uses loss signals from an agent's intuition model to encourage an agent to perform actions that will improve its intuition model (illustrated in Figure 1.1). Our contribution in this paper is twofold: (1) we propose a graphical physics network that can extract physical relations between objects and predict their physical behaviors in a 3D environment, and (2) we integrate the graphical physics network with the deep reinforcement learning network where we introduce the intrinsic reward normalization method that encourages the agent to efficiently explore actions and expedite the improvement of its intuition model. The results show that our actor network is able to perform a wide set of different actions and our prediction network is able to predict object's change in velocity and position very accurately.

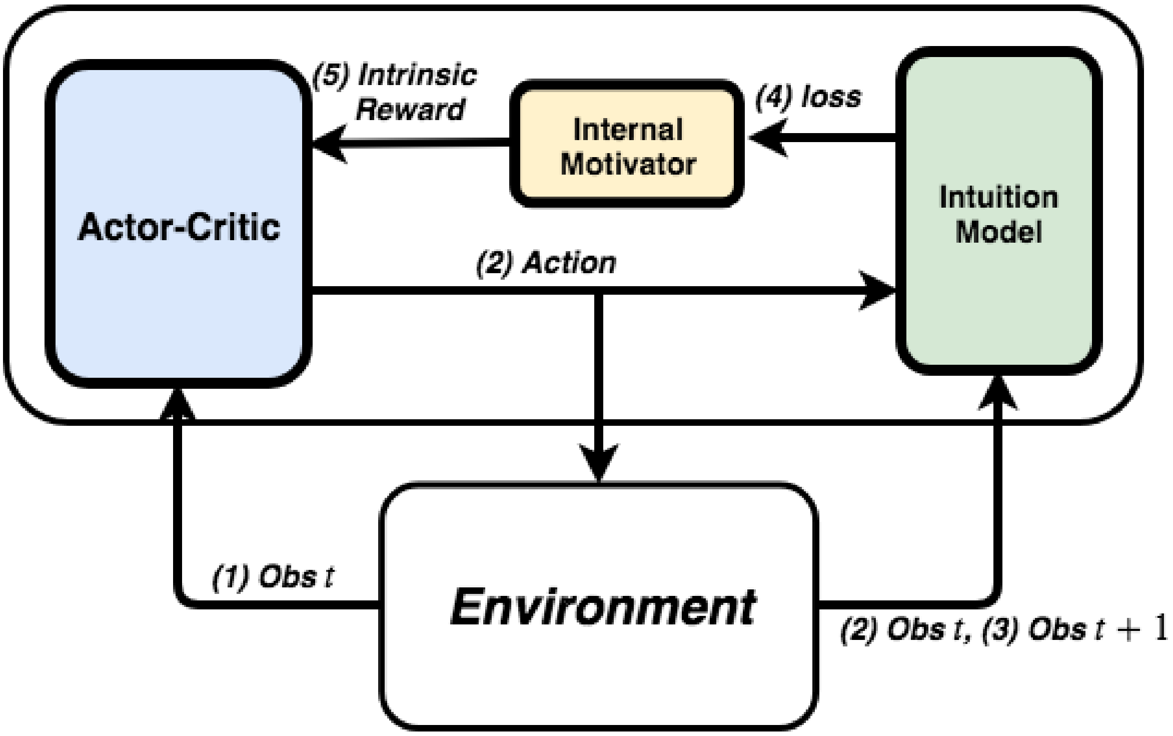

Figure 1.1: This diagram illustrates how an agent chooses an action based on loss incurred from its predictor which tries to mimic the behavior of an environment (or a subset of an environment). (1) After making an observation at time t, the agent's actor module chooses an action. (2) The predictor module takes the action and the observation at time t and makes a prediction; (3) the predictor module then compares its prediction to the observation from the environment at $t + 1$ and (4) outputs a loss. (5) The loss value is converted into an intrinsic reward by the internal motivator module and is sent to the actor's replay buffer to be stored for future training.

Chapter 2. Approach

In this section, we explain representations of objects and their relations, followed by our internal motivator.

2.0.1 Representation

We represent a collection of objects as a graph where each node denotes an object and each edge represents a pairwise relation of objects. Due to its simple structure, we use sphere as our primary object throughout this paper. Given N different objects, each object's state, s_{obj_i}, consists of its features, d_{obj_i}, and its relations with other objects:

$$d_{obj_i} = [x, y, z, vx, vy, vz, Fx, Fy, Fz, r, m] \,, \tag{2.1}$$

$$r_{ij} = f_r(d_{obj_i}, d_{obj_j}) \,, \tag{2.2}$$

$$\mathbf{R}_{obj_i} = \begin{bmatrix} r_{i1} & r_{i2} & \cdots & r_{iN} \end{bmatrix}^T \,, \tag{2.3}$$

$$e_{obj_i} = \sum_{i \neq j, j \in [1,N]} r_{ij} \,, \tag{2.4}$$

$$s_{obj_i} = [d_{obj_i}, e_{obj_i}] \,, \tag{2.5}$$

where d_{obj_i} contains object's Euclidean position (x, y, z), Euclidean velocity (vx, vy, vz), Euclidean force (Fx, Fy, Fz), size r and mass m. Some of these features are not readily available upon observation, and some works have addressed this issue by using convolutional neural networks [Lerer et al.(2016)Lerer, Gross, and Fergus, Watters et al.(2017)Watters, Zoran, Weber, Battaglia, Pascanu, and Tacchetti, Fragkiadaki et al.(2015)Fragkiadaki, Agrawal, Levine, and Malik]; in this paper, we assume that these features are known for simplicity. To provide location invariance, we use an object-centric coordinate system, where the xy origin is the center of the sphere's initial position and the z origin is the object's initial distance to a surface (e.g. floor).

Relation Encoder An object's state also includes its pairwise relations to other objects. A relation from object i to object j is denoted as r_{ij} and can be computed from the relation encoder, f_r, that takes two different object features, d_{obj_i} and d_{obj_j}, and extracts their relation r_{ij} (shown in Eq. (2.2)). Note that r_{ij} is directional, and $r_{ij} \neq -r_{ji}$. Once all of object's pairwise relations are extracted, they are stored in a matrix, $\mathbf{R}_{obj_i}$. The sum of all pairwise relations of an object yields e_{obj_i}, and when concatenated with the object feature d_{obj_i}, we get the state of the object s_{obj_i}.

An observation is a collection of every object's state and its pairwise relations:

$$obs = [(s_{obj_1}, \mathbf{R}_{obj_1}), (s_{obj_2}, \mathbf{R}_{obj_2})), \cdots, (s_{obj_n}, \mathbf{R}_{obj_n})] \,.$$

2.0.2 Object-based Attention

When an agent performs an action on an object, two things happen: (1) the object on which action was performed moves and (2) some other object (or objects) reacts to the moving object. Using the terms defined by [Chang et al.(2016)Chang, Ullman, Torralba, and Tenenbaum], we call the first object **focus object** and the other object **relation object**. Given the focus object and the relation object, we associate agent's action to the observation of both focus object and relation object. The agent's job is to monitor the physical behaviors of both objects when selected action is performed and check whether its action elicited intended consequences.

2.0.3 Internal Motivator

ATARI games and other reinforcement learning tasks often have clear extrinsic rewards from the environment, and the objective of reinforcement learning is to maximize the extrinsic reward. In contrast, intuition model learning does not have any tangible reward from the external environment. For example, human infants are not extrinsically rewarded by the environment for predicting how objects move in the physical world.

In order to motivate our agent to continuously improve its model, we introduce an internal motivator module. Similar to the work of [Pathak et al.(2017)Pathak, Agrawal, Efros, and Darrell], the loss value from an intuition model of an agent is converted into a reward by the internal motivator module ϕ:

$$r_t^{intrinsic} = \phi(loss_t)\,, \tag{2.6}$$

where $r_t^{intrinsic}$ is the intrinsic reward at time t, and $loss_t$ is loss from a prediction network at time t.While not having any extrinsic reward may result in ad hoc intrinsic reward conversion methods, we show that our model actually benefits from our simple, yet effective intrinsic reward normalization method. Note that while our method adopted to use the prediction error as the loss based on a prior work [Pathak et al.(2017)Pathak, Agrawal, Efros, and Darrell, **?**], our approach mainly focuses on constructing an intuitive physics model, while the prior approaches are designed for the curiosity-driven exploration in a game environment.

Intrinsic Reward Normalization In conventional reinforcement learning problems, the goal of an agent is to maximize the total reward. To do so, it greedily picks an action with the greatest Q value. If that action does indeed maximize the overall reward, the Q value of that action will continue to increase until convergence. However, sometimes this could cause an agent to be stuck in a sub-optimal policy in which the agent is oblivious to other actions that could yield a greater total reward.

To ameliorate the problem of having a sub-optimal policy, the most commonly used method is a simple ϵ-greedy method, which randomly chooses an action with ϵ probability. This is a simple solution, but resorts to randomly picking an action for exploration. An efficient way for exploration in stationary state setting with discrete actions is to use the upper confidence bound (UCB) action selection [Sutton & Barto(1998)Sutton and Barto]:

$$a_t = \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right],$$

where $Q_t(a)$ is the Q value of action a at time t, $N_t(a)$ is the number of times action a was chosen. However, this method has several shortcomings: (1) it is difficult to be used in non-stationary states since UCB does not take states into account, and (2) it can be more complex than ϵ-greedy method since it needs to keep track of the number of actions taken. Notice that this heuristic assumes that Q value of some action a at time t, or $Q_t(a)$, converges to some upper bound $\mathcal{U} \in \mathbb{R}$; i.e., $Q_t(a) \to \mathcal{U}^-$ as $t \to \infty$.

Intuitive physics learning is, however, slightly different from that of conventional reinforcement learning in that $\lim_{t\to\infty} Q_t(a) = 0 \;\; \forall a$. To show this, assuming that our intuition model improves in accuracy, it is equivalent to saying that loss is decreasing: $loss_t \to 0$ as $t \to \infty$. Earlier, we defined our intrinsic reward $r_t^{intrinsic} = \phi(loss_t)$ at Eq. (2.6). Since $loss_t \to 0$, we know that $r_t^{intrinsic} \to 0$ as long as ϕ is a

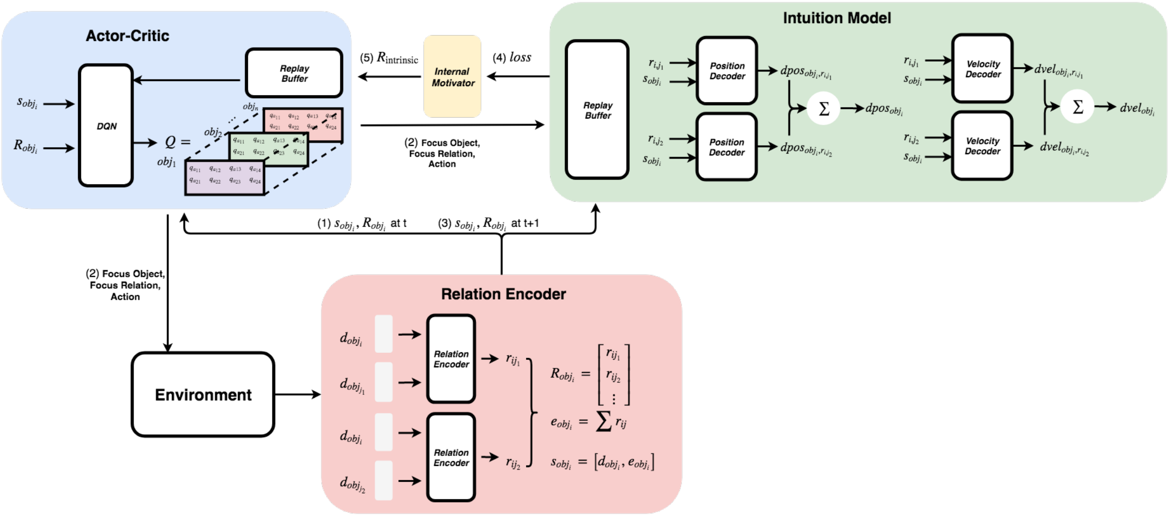

Figure 2.1: This shows a detailed diagram of our overall approach shown in Fig. 1.1. (Bottom Right) For every pair of objects, we feed their features into our relation encoder to get relation r_{ij} and object i's state s_{obj_i}. (Top Left) Using the greedy method, for each object, we find the maximum Q value to get our focus object, relation object, and action. (Top Right) Once we have our focus object and relation object, we feed their states and all of their relations into our decoders to predict the change in position and change in velocity.

continuous and increasing function. In both stationary and non-stationary state problems, if we train $Q_t(a)$ to be $r_t^{intrinsic}$, we can show that $\lim_{t\to\infty} Q_t(a) = 0$. However, if we train $Q_t(a)$ in an infinite state space, we cannot prove that $Q_t(a) \to 0)$ due to the complicated nature of asymptotic analysis Am I right?. I'm not sure. This is a bit more complex. I'd prefer just not mentioning it for now and come back later

In order to capitalize on this property, we first normalize $\phi(loss_t)$ by some upper bound $\mathcal{U}$ to get a normalized reward:

$$R_t^{intrinsic} = \frac{\phi(loss)}{\mathcal{U}}\,, \tag{2.7}$$

which restricts $R_t^{intrinsic} \in [0, 1]$. Additionally, we normalize all $Q_t(a)$ by using the following equation, which is also known as the Boltzmann distribution of actions [Sutton & Barto(1998)Sutton and Barto]:

$$\mathcal{Q}_t(a) = \left[\frac{e^{Q_t(a)}}{\sum_{k=1}^{|A|} e^{Q_t(k)}}\right], \tag{2.8}$$

$$A_t = \arg\max_{a'} \mathcal{Q}_t(a')\,, \tag{2.9}$$

where $\mathcal{Q}_t(a)$ is the normalized Q value of action a at time t, $|A|$ is the cardinality of the discrete action space, and A_t is the action with the highest, normalized Q value.

Given $\mathcal{Q}_t(a) \in [0, 1]$ and $R_t^{intrinsic} \in [0, 1]$, as we train $\mathcal{Q}_t(a)$ with $R_t^{intrinsic}$ using gradient descent, this will naturally increase $Q_t(a)$ of actions that have not been taken according to our normalization step (Eq. (2.8)); i.e., $Q_t(a')$ of a' actions that have been taken decreases and thus other actions that were not taken will have relatively bigger Q values.

Compared to other methods such as UCB, our method has the benefit of not needing to explicitly keep track of the number of actions taken, and instead takes advantage of the decreasing behavior of loss values to encourage exploration. Moreover, because our method does not keep track of actions, it can be used in non-stationary state problems as well, which we show later in Sec. ??.

2.0.4 Replay Buffers

Our agent has two separate replay buffers: one to store object's reward values to train the actor (**actor replay buffer**) and another to store the physical behaviors of objects to train the intuitive physics model (**prediction replay buffer**). For both networks, past experiences are sampled uniformly despite the fact that human brains are able to selectively retrieve past experiences from memory.

Despite uniform sampling, as our agent continuously experiments with objects, both replay buffers are more likely to contain experiences that the agent predicted with low accuracy. This is because our agent will greedily choose an action with the greatest Q value as shown in Eq. (2.9) and by design, action with the greatest Q value also has the greatest loss (equivalently low accuracy) by Eq. (2.7). Nonetheless, if the replay buffer is not big enough, this could let the agent overfit its intuitive physics model to experiences with high stochasticity. An ideal solution is to let agent curate its replay buffer content to find a set of experiences that can maximize the generalizability of the network. There are works that have addressed similar issues such as prioritized replay buffer [**?**]. However, we use uniform sampling in our work for its simplicity.

Chapter 3. Model

Using deep neural networks, our agent network can be separated into three different sub-networks: relation encoders, deep Q network, and position & velocity decoders, illustrated in Fig. 2.1.

3.0.1 Deep Q Network

Inspired by the recent advances in deep reinforcement learning [Mnih et al.(2015)Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, and etc, Schulman et al.(2015)Schulman, Levine, Moritz, Jordan, and Abbeel, Schulman et al.(2017)Schulman, Wolski, Dhariwal, Radford, and Klimov, Silver et al.(2014)Silver, Lever, Heess, Degris, Wierstra, and Riedmiller], we use the object oriented [Diuk et al.(2008)Diuk, Cohen, and Littman] deep Q network to choose three things: (1) focus object, (2) relation object, and (3) action.

For each object, our deep Q network takes $\mathbf{R}_{obj_i}$ and s_{obj_i} as input and computes $(N-1) \times |A|$ matrix, whose column represents action Q values and row represents obj_i's relation object. Computing this for all objects, the final output of the network has a shape of $N \times (N-1) \times |A|$, which we call $\mathcal{Q}$, shown in the top left module

in Fig. 2.1. Our agent greedily finds focus object, relation object and action:

$$\textit{focus object, relation object, action} = \arg\max_{i,r,a} \mathcal{Q}_{i,r,a} \,,$$

where i indicates the focus object index, r is the relation object index, and a denotes the action index.

Our deep Q network does not use a target network, and the actor replay buffer samples experiences of every object, instead of randomly sampling from all experiences uniformly. This is done to prevent an agent from interacting with only one object and from generalizing the behavior of one object to other objects.

With this setup, we set our stationary state target Q value, y_t^s, to be:

$$y_t^s = R_{intrinsic}^t \,,$$

and non-stationary state target Q value, y_t^{ns}, to be

$$y_t^{ns} = \begin{cases} R_t^{intrinsic} & \text{if reset occurs at t+1} \\ \min(1, (R_t^{intrinsic} + \gamma \max_{a'} Q_{a'}(s_{obj_i}^{t+1}))) & \text{o.w.} \end{cases} \,, \tag{3.1}$$

where γ is a discount factor in $[0,1]$ and objects' states reset when one of the objects goes out of bounds. We provide details of non-stationary state experiment and bounds in Sec. 4. For stationary state problems, since there is only single state, we only use $R_t^{intrinsic}$ to update our Q value. For non-stationary problems, we take the subsequent state into account and update the Q value with the sum of reward and discounted next state Q value. Note that our Q values and rewards reside in $[0,1]$ because of the reward normalization method; therefore, when the sum of reward and discounted next state reward exceeds 1, we clip the target value to 1.

3.0.2 Position & Velocity Decoders

The predicted position and velocity of each object is estimated by the predictor module which is placed inside a green box in Fig. 2.1).

An object's state, s_{obj_i} and all of its pairwise relations, $\mathbf{R}_{obj_i}$, are fed into both position and velocity decoders to predict the change in position and change in velocity of obj_i. For each pairwise relation, r_{ij}, we get an output $dpos_{i,r_{ij}}$ from the position decoder and $dvel_{i,r_{ij}}$ from the velocity decoder.

Once all relations are accounted for, the sum of all $dpos_{i,r_{ij}}$ and $dvel_{i,r_{ij}}$ are the final predicted change in position, $dpos_{obj_i}$, and predicted change in velocity, $dvel_{obj_i}$, of an object i:

$$\begin{aligned} dpos_{obj_i} &= \sum_{i \neq j, j \in [1,N]} dpos_{obj_i, r_{i,j}} \, , \\ dvel_{obj_i} &= \sum_{i \neq j, j \in [1,N]} dvel_{obj_i, r_{i,j}} \, , \end{aligned}$$

We train both decoders and relation encoder with the sum of mean squared errors of positions and velocities:

$$\begin{aligned} loss = \sum_{k=\{i,r\}} &||dpos_{obj_k} - dpos'_{obj_k}||_2 \\ &+ ||dvel_{obj_k} - dvel'_{obj_k}||_2 \, , \end{aligned}$$

where i is the focus object index and r is the relation object index. $dpos'_{obj_k}$ and $dvel'_{obj_k}$ are the ground truth change in position and velocity of an object, and are readily available by the physics engine.

Chapter 4. Experiment Setup

Objects In our experiment, we use spheres as primary objects due to its simple structure (i.e. we can represent an object with only one variable - radius). We use the center of the sphere as its xy position and its distance to a surface, i.e. floor, as z position. We used pybullet as our physics simulator with timestep of 1/240 seconds with 30 fps. As shown in Figure 4.1, we use three different scenes: 3-object, 6-object, and 8-object scenes. Objects are color coded so that each denotes different weight: red is 1kg, green is 0.75kg, blue is 0.5kg, and white is 0.25kg. Each object can have radius of 5cm or 7.5cm.

Normalized Action Space We provide an agent with actions in three different directions: x, y and z. We experiment with a set of 27 actions ($x, y \in \{-1.0, 0.0, 1.0\}$ and $z \in \{0.0, 0.75, 1.0\}$), and a set of 75 actions ($x, y \in \{-1.0, -0.5, 0.0, 0.5, 1.0\}$ and $z \in \{0.0, 0.75, 1.0\}$). An action chosen from the Q network is then multiplied by the max force, which we set to 400N.

Performance Metric Unlike conventional reinforcement learning problems, our problem does not contain any explicit reward that we can maximize on. For that reason, there is no clear way to measure the performance of our work. Outputs from both prediction network and deep Q network rely heavily on how many different actions the agent has performed. For instance, if an agent performs only one action, the prediction loss will converge to 0 very quickly, yet agent would have only learned to predict the outcome of one action. Therefore, we provide the following performance metrics to see how broadly our agent explores different actions and how accurately it predicts outcomes of its actions:

- **Action Coverage** We measure the percentage of actions covered by an agent. There are in total of $N \times (N-1) \times |A|$ many actions for all pairs of objects. We use a binary matrix, M, to keep track of actions taken. We say that we covered a single action when an agent performs that action on all object relation pairs for every focus object. In short, if $\prod_{i,r} M_{i,r,a_k} = 1$ for some action a_k, then we say it covered the action a_k. Action coverage is then computed by $(\sum_a \prod_{i,r} M_{i,r,a})/|A|$. Action coverage value will tell us two things: (1) whether our predictor module is improving and (2) whether our agent is exploring efficiently. If the predictor module is not improving, the intrinsic reward associated with that action will not decrease, causing the agent to perform the same action repeatedly.

- **Prediction Error** Once an agent has tried all actions, we use prediction error to test whether agent's predictor module improved.

For other hyperparameters, we set the upper bound $\mathcal{U}$ to be infinity and ϕ to be an identity function.

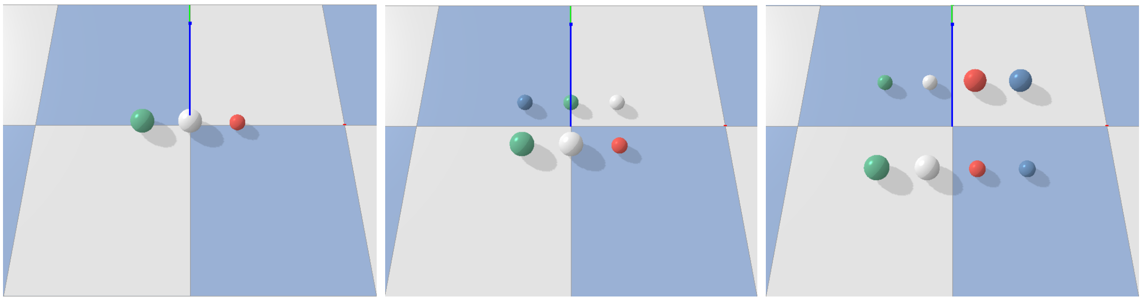

Figure 4.1: Experiments in three different scenes: 3-object, 6-object and 8-object scenes. Colors represent different weights where red is 1kg, green is 0.75kg, blue is 0.5kg and white is 0.25kg. The radius of each object can be either 5cm or 7.5cm. We experiment in two scenarios: stationary and non-stationary states.

Table 4.1: **Stationary state Action Coverage** We record the number of interactions our agent takes to perform actions in three different scenes. Each scene was experimented with two or more trials with different random seeds. We stop training when interaction count exceeds 75000 interactions.

	#Actions per relation	Total # Actions	Action Coverage	Interaction Count
3-Object Scene	27	162	1.0	770.38 ± 187.914
	75	450	1.0	2505.6 ± 401.968
6-Object Scene	27	810	1.0	7311.0 ± 2181.74
	75	2250	1.0	25958.67 ± 3609.25
8-Object Scene	27	1512	1.0	22836 ± 2225.0
	75	4200	0.8 ± 0.107	75000

Chapter 5. Experiments

In this section, we provide results of our experiments in stationary and non-stationary state problems.

5.0.1 Stationary State (Multi-armed Bandit Problem)

In stationary state, or multi-arm bandit problem, after an agent takes an action, we reset every object's position back to its original state. The initial states of objects in different scenes are shown in Figure 4.1. We test with two different action sets and compare the number of interactions it takes for an agent to try out all actions. We also test generalizability of our agent's prediction model to multiple objects.

Action Coverage As shown in Table 4.1, our agent successfully tries all available actions in 3-object and 6-object scenes for both set of actions. Despite the huge number of actions, our agent is able to intrinsically motivate itself to perform hundreds and thousands of actions. While there is no clear way to tell our method is the fastest, we are not aware of any previous work that measures the number of actions covered by an agent, since conventional reinforcement learning problems do not require an agent to perform all actions. The fastest way to cover all actions is to keep track of all actions and their Q values in a table; however, this method has scalability issues and cannot be extended to non-stationary state problems.

As number of actions increases, it, however, takes longer for our agent to cover all actions. In fact, for 8-object scene, it fails to achieve full action coverage when presented with 75 different actions per object. One possible explanation of this is the replay buffer. Using a sampling batch size of 1024, the actor replay buffer uniformly samples 1024 past experiences for all objects. For 8-object scene with 75 actions, each object has 525 unique actions per all pairs of objects. Compared to other scenes, the probability of getting 525 unique action experiences from 1024 samples is a lot lower, especially when the actor replay buffer has uneven distribution of actions. To make matters worse, our prediction replay buffer is limited in size. In our experiment, the prediction buffer of size $2.5e + 6$ saturates before our agent can perform all actions.. It is apparent that our approach cannot scale to scenes with more objects and actions with the current implementation of both actor and prediction replay buffers. We leave the improvement of replay buffers to future work.

Prediction Error To test whether our agent's intuitive physics model predicts object position and velocity correctly, we test it by computing the $L2$-norm between predicted and the actual position and velocity of object after one frame. Our agent's prediction model errors are plotted in Figure 5.2. For all scenes, the position errors of one frame prediction are within 0.002m and the velocity errors are within 0.15m/s. These errors quite small, given the fact that the objects in our scene can change its position from 0 to 0.18465m per frame and velocity change ranges from 0 to 6.87219m/s per frame.

Our prediction error could be reduced further with other network architectures. In fact, there are many works that are trying to develop better network models for intuitive physics learning. However, the aim of our work is to show that the loss value from any intuitive physics network can be converted into an intrinsic reward to motivate an agent to explore different actions. Observations from different actions result in a diverse set of training data, which can make the intuitive physics network more general and robust.

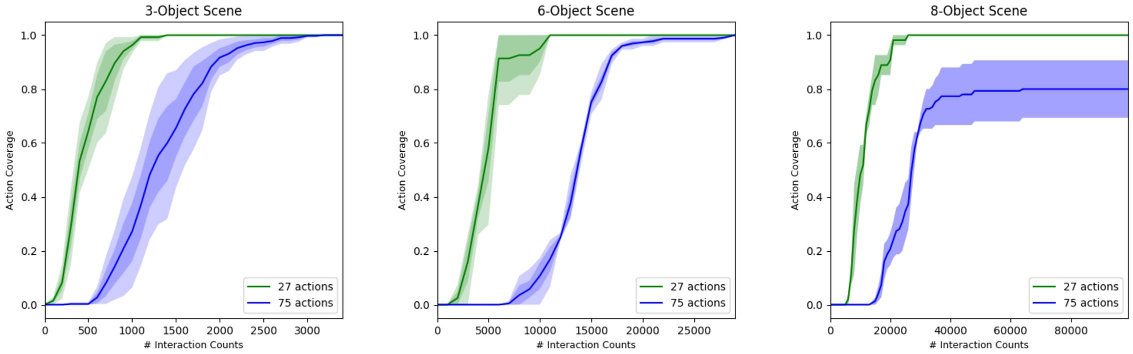

Figure 5.1: Action coverage results after 3 or more runs with different random seeds. (From left to right) Action coverage of our agent in 3-object, 6-object, and 8-object scene.

In generalization tasks, we allow our agent to apply forces on multiple objects and see if it can predict the outcome after one frame. We generate 100 experiments by randomly selecting focus objects and actions. We let our agent to predict the next position and velocity of all objects in the scene, and we measure the mean error of all predictions. Video of our results can be found in https://youtu.be/-18o6K5D4pQ. The results show that our agent's prediction network accurately predicts the physical behavior of colliding and non-colliding objects. Moreover, our agent's intuition model is able to generalize to multiple moving objects very well even though the agent was only trained with an observation of a pair of objects (i.e.focus and relation object). Our qualitative results show that the agent's prediction model is able to predict that collision causes a moving object to either deflect or stop, and causes idle objects to move. Additionally, despite not knowing about gravitational forces, it learns that objects fall when they are above a surface.

While there are other intuitive physics networks trained with supervised learning that can yield a higher accuracy, it is difficult for us to compare our results with theirs. The biggest reason is that supervised learning is provided with a well-defined set of training data a priori. In our work and reinforcement learning in general, data are collected by an agent, and those data are not identical to that of supervised learning, making it difficult to compare two different approaches in a fair setting. Additionally, the training process differs: supervised learning takes epoch based learning where it iterates over the same dataset multiple times until the network reaches a certain error rate on a validation set. In deep reinforcement learning, a small subset of data is randomly sampled from the replay buffer and is used to train the network on the fly.

5.0.2 Non-stationary State (Reinforcement Learning)

We extend our prediction model with deep Q network to non-stationary state problems where we do not reset the objects unless they go out of bounds. To increase the chance of collision, we provide 9 actions only on the xy place (i.e. $x, y \in \{-1, 0, 1\}$). We arbitrarily set the bounds to be $3m \times 3m$ square. Since there are no walls to stop objects from going out of bounds, objects have a low probability of colliding with another object. In order to make them collide on every interaction, we generated 51 test cases in which every interaction causes a collision among objects. Our agent's prediction model then predicts the object's location after varying number of frames (i.e. 1,2,4,10,15,30,45 frames).

Prediction Error Prediction network's errors are plotted in Figure 6.1. We see that when our agent predicts object locations after 1, 2, and 4 frames, the error is negligibly small. However, our agent is uncertain when predicting object location after 10 or more frames. This is because the error from each frame accumulates and causes objects to veer away from the actual path. Our qualitative results of non-stationary problem can be found in https://youtu.be/OpxBMYqeq70. Similar to stationary state results, our agent's prediction network accurately predicts an object's general direction and their movements, despite having infinitely many states. The agent's prediction network is able to predict whether an object will stop or deflect from its original trajectory when colliding with another object. Even if the agent's prediction network fails to predict the correct position of a moving object, it still makes a physically plausible prediction.

Limitation We would like to point out that although our agent performed thousands of interactions, our agent still fails to learn that objects do not go through one another, as seen in our qualitative results. This is very noticeable when objects are moving fast. We conjecture that once our agent makes a wrong prediction, the predicted object overlaps with another object, causing our agent fails to predict subsequent positions and velocities. One plausible explanation is that it has never seen such objects overlap in its training data, hence fails to predict it accurately.

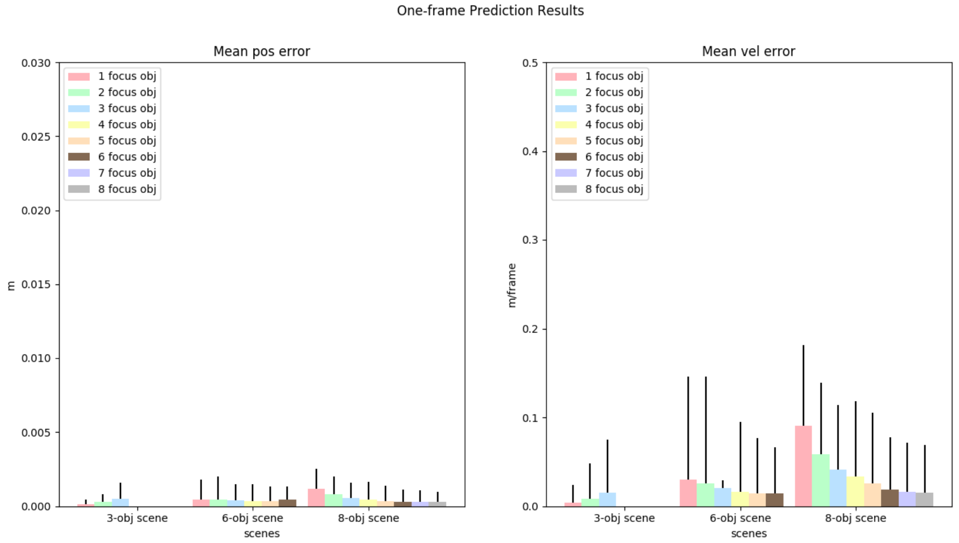

Figure 5.2: Mean position and velocity prediction errors after 1 frame with different number of focus objects in 3-object, 6-object, and 8-object scene.

Chapter 6. Related Works and Contributions of this Dissertation

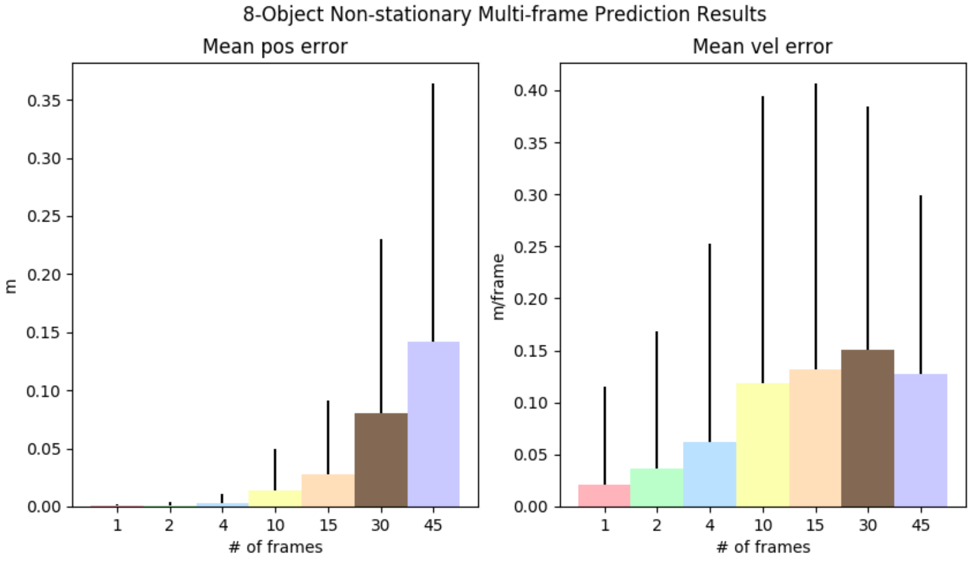

Figure 6.1: We test our intuition model in the non-stationary problem. Fewer prediction steps incur almost no error, while higher prediction steps, i.e. more than 4 frames, incur a high loss due to cumulative errors.

6.0.1 Deep Reinforcement Learning

Recent advances in deep reinforcement learning have achieved super-human performances on various ATARI games [Mnih et al.(2015)Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, and etc] and robotic control problems [Schulman et al.(2015)Schulman, Levine, Moritz, Jordan, and Abbeel, Schulman et al.(2017)Schulman, Wolski, Dhariwal, Radford, and Klimov, Silver et al.(2014)Silver, Lever, Heess, Degris, Wierstra, and Riedmiller]. While these approaches have achieved state-of-the-art performances on many tasks, they are often not easily transferable to other tasks because these networks are trained on individual tasks.

As opposed to model-free methods, model-based approaches create a model of an environment, which equips agents with the ability to predict and plan. Dyna-Q, proposed by [Sutton(1990)], integrated model-free with model-based learning so an agent can construct a model of an environment, react to the current state and plan actions by predicting future states. More recent work in model based reinforcement learning [Oh et al.(2017)Oh, Singh, and Lee] proposed a value prediction network that learns a dynamic environment and predicts future values of abstract states conditioned on options.

6.0.2 Intrinsic Motivation and Curiosity

Early work by [Berlyne(1966)] showed that both animals and humans spend a substantial amount of time exploring that is driven by curiosity. Furthermore, Berlyne's theory suggests that curiosity, or intrinsic motivation [Barto et al.(2004)Barto, Singh, and Chentanez, Chentanez et al.(2005)Chentanez, Barto, and Singh], is triggered by novelty and complexity.

The idea of integrating curiosity, and its counterpart boredom, with reinforcement learning was suggested by [Schmidhuber(1991)], and showed that intrinsic reward can be modeled to motivate agents to explore areas with high prediction errors [Schmidhuber(2010)]. Using deep learning, [Pathak et al.(2017)Pathak, Agrawal, Efros, and Darrell] proposed an intrinsic curiosity module that outputs a prediction error in the state feature space, which is used as an intrinsic reward signal. Our work adopted this approach of using the prediction error for our intrinsic reward.

6.0.3 Intuitive Physics

At an early age, human infants are equipped with a "starter pack" [Lake et al.(2017)Lake, Ullman, Tenenbaum, and Gershman], which includes a sense of intuitive physics. For instance, when observing a moving ball, our intuitive physics can sense how fast the ball is going and how far the ball will go before it comes to a complete halt. This intuitive physics is present as a prior model and accelerates future learning processes. Works done by [Battaglia et al.(2013)Battaglia, Hamrick, and Tenenbaum] and [Hamrick(2011)] show that humans have an internal physics model that can predict and influence their decision making process.

Recent works have focused on using deep learning to model the human's intuitive physics model. [Lerer et al.(2016)Lerer, Gross, and Fergus] used a 3D game engine to simulate towers of wooden blocks and introduced a novel network, PhysNet, that can predict whether a block tower will collapse and its trajectories. [Fragkiadaki et al.(2015)Fragkiadaki, Agrawal, Levine, and Malik] proposed a visual predictive model of physics where an agent is able to predict ball trajectories in billiards. Another work by [Chang et al.(2016)Chang, Ullman, Torralba, and Tenenbaum] proposed a neural physics engine that uses an object based representation to predict the state of the focus object given a physical scenario. [Battaglia et al.(2016)Battaglia, Pascanu, Lai, Rezende, and kavukcuoglu] presented an interaction network that combined structured models, simulation, and deep learning to extract relations among objects and predict complex physical systems. We extend the previous works by integrating deep reinforcement learning that intrinsically motivates our agent to improve its physics model.

Chapter 7. Concluding Remarks

In this paper, we have proposed a graphical physics network integrated with deep Q learning and a simple, yet effective reward normalization method that motivates agents to explore actions that can improve its model. We have demonstrated that our agent does indeed explore most of its actions, and our graphical physics network is able to efficiently predict object's position and velocity. We have experimented our network on both stationary and non-stationary problems in various scenes with spherical objects with varying masses and radii. Our hope is that these pre-trained intuition models can later be used as a prior knowledge for other goal oriented tasks such as ATARI games or video prediction.

Bibliography

[Baillargeon(2007)] Baillargeon, Renée. *The Acquisition of Physical Knowledge in Infancy: A Summary in Eight Lessons*, chapter 3, pp. 47–83. Wiley-Blackwell, 2007. ISBN 9780470996652. doi: 10.1002/9780470996652.ch3.

[Barto et al.(2004)Barto, Singh, and Chentanez] Barto, A. G., Singh, S., and Chentanez, N. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of International Conference on Developmental Learning (ICDL)*. MIT Press, Cambridge, MA, 2004.

[Battaglia et al.(2016)Battaglia, Pascanu, Lai, Rezende, and kavukcuoglu] Battaglia, Peter, Pascanu, Razvan, Lai, Matthew, Rezende, Danilo Jimenez, and kavukcuoglu, Koray. Interaction networks for learning about objects, relations and physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 4509–4517, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9.

[Battaglia et al.(2013)Battaglia, Hamrick, and Tenenbaum] Battaglia, Peter W., Hamrick, Jessica B., and Tenenbaum, Joshua B. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013. ISSN 0027-8424. doi: 10.1073/pnas.1306572110.

[Berlyne(1966)] Berlyne, D. E. Curiosity and exploration. *Science*, 153(3731):25–33, 1966. ISSN 0036-8075. doi: 10.1126/science.153.3731.25.

[Chang et al.(2016)Chang, Ullman, Torralba, and Tenenbaum] Chang, Michael B, Ullman, Tomer, Torralba, Antonio, and Tenenbaum, Joshua B. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[Chentanez et al.(2005)Chentanez, Barto, and Singh] Chentanez, Nuttapong, Barto, Andrew G., and Singh, Satinder P. Intrinsically motivated reinforcement learning. In Saul, L. K., Weiss, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 17*, pp. 1281–1288. MIT Press, 2005.

[Diuk et al.(2008)Diuk, Cohen, and Littman] Diuk, Carlos, Cohen, Andre, and Littman, Michael L. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 240–247, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390187.

[Fragkiadaki et al.(2015)Fragkiadaki, Agrawal, Levine, and Malik] Fragkiadaki, Katerina, Agrawal, Pulkit, Levine, Sergey, and Malik, Jitendra. Learning visual predictive models of physics for playing billiards. *CoRR*, abs/1511.07404, 2015. URL http://arxiv.org/abs/1511.07404.

[Hamrick(2011)] Hamrick, Jessica. Internal physics models guide probabilistic judgments about object dynamics. 01 2011.

[Lake et al.(2017)Lake, Ullman, Tenenbaum, and Gershman] Lake, Brenden M., Ullman, Tomer D., Tenenbaum, Joshua B., and Gershman, Samuel J. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 2017.

[Lerer et al.(2016)Lerer, Gross, and Fergus] Lerer, Adam, Gross, Sam, and Fergus, Rob. Learning physical intuition of block towers by example. In Balcan, Maria Florina and Weinberger, Kilian Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 430–438, New York, New York, USA, 20–22 Jun 2016. PMLR.

[Mnih et al.(2015)Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, and etc] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin A., and etc. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

[Oh et al.(2017)Oh, Singh, and Lee] Oh, Junhyuk, Singh, Satinder, and Lee, Honglak. Value prediction network. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*. 2017.

[Pathak et al.(2017)Pathak, Agrawal, Efros, and Darrell] Pathak, Deepak, Agrawal, Pulkit, Efros, Alexei A., and Darrell, Trevor. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[Schmidhuber(2010)] Schmidhuber, J. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, Sept 2010. ISSN 1943-0604. doi: 10.1109/TAMD.2010.2056368.

[Schmidhuber(1991)] Schmidhuber, Jürgen. A possibility for implementing curiosity and boredom in model-building neural controllers, 1991.

[Schulman et al.(2015)Schulman, Levine, Moritz, Jordan, and Abbeel] Schulman, John, Levine, Sergey, Moritz, Philipp, Jordan, Michael, and Abbeel, Pieter. Trust region policy optimization. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1889–1897. JMLR.org, 2015.

[Schulman et al.(2017)Schulman, Wolski, Dhariwal, Radford, and Klimov] Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[Silver et al.(2014)Silver, Lever, Heess, Degris, Wierstra, and Riedmiller] Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. I–387–I–395. JMLR.org, 2014.

[Sutton(1990)] Sutton, Richard S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *In Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224. Morgan Kaufmann, 1990.

[Sutton & Barto(1998)Sutton and Barto] Sutton, Richard S. and Barto, Andrew G. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

[Watters et al.(2017)Watters, Zoran, Weber, Battaglia, Pascanu, and Tacchetti] Watters, Nicholas, Zoran, Daniel, Weber, Theophane, Battaglia, Peter, Pascanu, Razvan, and Tacchetti, Andrea. Visual interaction networks: Learning a physics simulator from video. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4539–4547. Curran Associates, Inc., 2017.

Curriculum Vitae

Name	:	Jae Won Choi
Date of Birth	:	April 07, 1993
Birthplace	:	Seoul, Republic of Korea
Address	:	Daejeon

Educations

2008. 9. – 2012. 5.	Williston Northampton School (High School)
2012. 8. – 2015. 12.	Carnegie Mellon University (Electrical and Computer Engineering) (BS)
2016. 12. – 2019. 2.	Korea Advanced Institute of Science and Technology (Robotics) (MS)

Career

2013. 8. – 2016. 5.	Undergraduate Teaching Assistant (more than 3 classes)