

석사학위논문
Master's Thesis

적대적 생성신경망을 통한 이종 행위자 스타일의
모방학습

Imitation Learning for Different Player Style using Generative
Adversarial Networks

2018

김도형 (金度亨 Kim, Do-Hyeong)

한국과학기술원

Korea Advanced Institute of Science and Technology

석사학위논문

절대적 생성신경망을 통한 이종 행위자 스타일의
모방학습

2018

김도형

한국과학기술원

산업 및 시스템 공학과

적대적 생성신경망을 통한 이종 행위자 스타일의 모방학습

김 도 형

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2017년 12월 28일

심사위원장 문 일 철

심사위원 박진규

심사위원 윤세영

Imitation Learning for Different Player Style using Generative Adversarial Networks

Do-Hyeong Kim

Advisor: Il-Chul Moon

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 Industrial and Systems Engineering

Daejeon, Korea
2017, December

Approved by

Il-Chul Moon
Professor of Industrial Systems Engineering

The study was conducted in accordance with Code of Research Ethics¹.

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.

MIE 20163077	김도형. 적대적 생성신경망을 통한 이종 행위자 스타일의 모방학습. 산업 및 시스템 공학과 . 2018년. 33+iii 쪽. 지도교수: 문일철. (한글 논문) Do-Hyeong Kim. Imitation Learning for Different Player Style using Generative Adversarial Networks. Department of Industrial and Systems Engineering . 2018. 33+iii pages. Advisor: Il-Chul Moon. (Text in Korean)
-----------------	---

초 록

실생활 예시에서 쉽게 찾을 수 있는 문제들은 대개 순차적 의사 결정 문제로 모델링 될 수 있으나, 강화학습 방법론을 사용하여 해당 문제를 해결하려 할 때 reward design이 어렵다는 단점을 가진다. Imitation learning의 경우, reward가 아닌 한정된 optimal action을 활용하여 최적 행동을 모방하여 순차적 의사 결정 문제를 해결한다. 본 연구에서는 최근 제안된 deep generative model인 VAEGAN의 구조를 활용한다. MDP로 정의된 순차적 의사 결정 문제에 대해 한정된 양의 state와 이에 해당되는 optimal action이 학습 데이터로 주어졌을 때, state의 정보를 효율적으로 함축하고 해당 정보를 최적 행동으로 재생성 할 수 있는 모델에 대해 제안하였다. 또, 개체 스타일에 대한 정보를 conditional하게 학습시켜 같은 state가 주어질 때 개체의 스타일에 맞는 행위를 생성할 수 있는 방법을 제시하였다.

핵심 낱말 Deep Generative Model, Imitation Learning, Variational Auto-encoder, Generative Adversarial Networks, Sequential Decision Problem

Abstract

Problems that can be easily found in real life examples can usually be modeled as sequential decision problems. However they have the disadvantage that reward design is difficult when trying to solve problems using reinforcement learning methodology. In the case of Imitation learning, sequential decision problems are solved by imitating optimal behaviors using limited optimal action rather than reward. In this study, we modified the structure of VAEGAN, which is a recently developed deep generative model, to obtain the information of state efficiently when a limited amount of states and optimal action as learning data. and suggests a model that can reproduce the information as an optimal behavior. In addition, we proposed a method to conditionally learn the information about the object style and to create an action for each purpose under the same state.

Keywords Deep Generative Model, Imitation Learning, Variational Auto-encoder, Generative Adversarial Networks, Sequential Decision Problem

차례

차례	ii
표 차례	iii
그림 차례	iii
제 1 장 Introduction	1
1.1 Research Background and Objective	1
1.2 Thesis Structure	2
제 2 장 Previous Research	3
2.1 Deep Generative Model	3
2.1.1 Variational Auto-encoder; VAE	3
2.1.2 Generative Adversarial Network; GAN	4
2.1.3 VAEGAN	5
2.2 Conditional Deep Generative Model	6
2.2.1 Conditional Variational Auto-encoder	6
2.2.2 Conditional Generative Adversarial Network	7
제 3 장 Methodology	8
3.1 Network Overview	8
3.1.1 Encoder	9
3.1.2 Generator	10
3.1.3 Discriminator	11
제 4 장 Results	12
4.1 ObjectWorld Benchmark Dataset	12
4.1.1 MDP formalism	12
4.1.2 Reward Setting: ObjectWorld	13
4.1.3 Policy generation	15
4.1.4 Generate Dataset for the Network Training & Testing	15
4.2 Experimental Setting	18
4.3 Quantitative Results	18
4.4 Qualitative Results	19
제 5 장 Discussion	21
5.1 Improvement Direction	21

5.1.1	ObjectWorld Dataset	21
5.1.2	Network Structure & Parameters	23
5.2	Experiment for Improvement	23
5.2.1	State Input Information	23
제 6 장	CONCLUSION AND FUTURE WORK	27
참 고 문 헌		29
약 력		31

표 차례

2.1	Table of Notation	3
4.1	ObjectWorld Model Parameters	17
4.2	Data Statistics	17
4.3	Model Parameter	18

그림 차례

2.1	Graphical model of VAE	4
2.2	Structure of GAN	4
2.3	Structure of VAE/GAN	5
2.4	Structure of CVAE	6
2.5	Structure of CGAN	7
3.1	Model Overview	8
3.2	Encoder Network	9
3.3	Generator Network	10
3.4	Discriminator Network	11
4.1	Object World Benchmark Model: Concept	13
4.2	Positive reward condition	13
4.3	Negative reward condition	14
4.4	Reward construction for given ObjectWorld	14
4.5	Different reward generation from different agent style	15
4.6	Data generation of state	16
4.7	Data generation of reward, style, and action	17
4.8	Cumulative reward during 20 policy propagation for generated and optimal action	19
4.9	Generated action & Optimal Policy for same state. (Top: Epoch 10, Middle: Epoch 50, Bottom: Epoch 100)	20
5.1	Distance from nearest object	22
5.2	Experiment loss result: Full(Inner/outer) feature	25
5.3	Experiment loss result: Part(outer) feature	26

제 1 장 Introduction

1.1 Research Background and Objective

실 사회에서 발생하는 문제들을 공학적으로 모델링하고자 할 때, 순차적 의사 결정 문제로 모델링 될 수 있는 예시는 주변에서 쉽게 찾아볼 수 있다. 이러한 순차적 의사 결정 문제를 해결하는 대표적인 방법론으로는 강화학습(Reinforcement learning)이 존재한다. 강화학습은 특정 환경 (Environment)에서 지정된 행동 (Action)을 수행하도록 정의된 개체(Agent)가 존재할 때, 개체가 관찰 가능한 정보 (Observation)를 인식하고, 현재 상태 (State)를 파악한 뒤 수행할 수 있는 행동 가운데 누적된 보상값(Reward)을 최대로 하는 최적 행동 (Optimal action)을 선택할 수 있도록 학습시키는 방법이다. 그러나 강화학습 방법론을 현실상 제시되는 문제 상황에 적용시키는 것은 쉽지 않다. 강화학습의 경우, 그 학습의 성능이 주어진 문제 상황에 적합한 보상함수인지의 여부가 매우 중요하게 작용하기 때문에 보다 정확한 보상함수의 정의가 필요하다. 하지만 해당 문제 상황에 영향을 끼칠 수 있는 요소 중 곁으로 쉽게 드러나지 않는 요소들이 존재할 수 있고, 또한 그러한 요소들이 모두 관찰될 수 있다 하더라도 각 요소가 어떤 방식으로 영향을 끼치는지, 그 중요도는 각 요소별로 어떻게 다른지 결정하는 것이 쉽지 않기 때문이다.

주어진 문제 상황을 마코브 결정 과정 (Markov Decision Process; MDP)[6]으로 정의하고, 정의된 MDP에 대한 보상함수를 역으로 추정하는 것은 모방학습(Imitation learning) 방법론 중 하나인 역강화학습 (Inverse reinforcement learning)[7]이 존재한다. 1998년 Russel이 처음 제안하였으며[12], 최근 Deep Learning의 발전에 따라 Fully Convolutional Networks[8]를 활용한 Maximum Entropy Deep Inverse Reinforcement Learning[9]등의 방법들이 지속적으로 해당 분야의 연구 성과로 산출되고 있다. 그러나, 역강화학습 방법론들의 고질적인 단점으로는 역강화학습 과정 속에 강화학습 과정이 반복적으로 진행되어야 하기에 학습이 느리고 효율적이지 못하다는 점이다. 이는 학습이 진행되고 있는 reward에 대해, 지속적으로 policy propagation을 진행하고 최적 행동 데이터(Expert trajectories)와 얼마나 떨어져 있는지를 지속적으로 관찰해야 하기에 발생한다.

모방학습의 또 다른 분야로는 행동복제(Behavioral cloning) 방법론이 존재한다. 행동복제 방법론은 특정 환경 내에서 최적 행동 개체의 행동 (optimal action) 데이터가 주어진다고 가정하고, 개체의 행동을 보상함수에 의해 결정하도록 학습하는 것이 아니라, 최적 행동 개체의 행동을 그대로 복제하는 방법론을 의미한다. 행동복제 방법론은 지도학습 (supervised learning) 방식으로 학습한다. 즉, Input data X 동일한

분포 $p(x)$ 로부터 sampled 되었다고 가정하고, sampled data x 에 대한 label data y 가 pair로써 존재하고 해당 pair들을 training data로 학습시켰을 때, 관찰되지 않은 또 다른 sampled data x' 에 대한 label을 예측할 수 있도록 학습시키는 것을 의미한다.

본 논문에서는 행동복제 방법론을 활용하여 MDP로 정의된 순차적 의사 결정 문제를 해결하고자 시도하였다. 해결하고자 시도한 순차적 의사 결정 문제는 역강화학습 분야에서 처음 제안된 ObjectWorld Benchmark Model로, grid-shaped된 environment에서 reward를 형성하는 object들이 존재할 때 정확한 reward에 대한 정보 없이, 관찰 가능한 object의 위치와 특성만을 input data로 하여 environment의 각 grid cell 위치에서 개체가 수행해야 할 행위를 예측하는 것을 목적으로 가진다. 유일하게 관찰 가능한 것이 object의 위치와 특성이나, object의 특성 중에는 개체의 행동에 영향을 끼치지 않는 요소 또한 관찰된다. 이러한 distractor가 존재하는 상황에서 최적의 행동을 찾을 수 있는 모델로써의 역할을 수행할 때 이는 본 모델의 핵심 기여부문이 된다. 더불어, 본 모델은 object의 위치와 특성이 같은 input data가 주어졌을 때, 다르게 행동할 수 있는 이종 행위자 스타일의 모방학습이 가능하도록 하는 목적 또한 가진다. 이는 동일한 input data가 들어오게 될 때, '어느 스타일의 개체의 행동을 산출할 것인가'에 대한 추가 data를 활용한 conditioning을 함으로써 가능하다.

1.2 Thesis Structure

두 번째 Chapter에서는 본 연구의 모델로써 활용되는 VAEGAN을 설명하고 해당 모델의 시초가 되는 deep generative model인 VAE와 GAN에 대해 설명한다. Chapter 3에서는 본 논문에서 제안하는 방법론 및 모델을 구성하는 network에 대해 서술한다. Chapter 4에서는 실험에서 사용되는 데이터인 ObjectWorld Model에 대해 설명하고, 실험 설정 및 결과에 대해 서술한다. Chapter 5에서는 본 모델의 한계점과 개선 방향 및 개선점에 대한 추가 연구 결과에 대해 서술하며, 마지막으로 Chapter 6에서는 본 논문의 결론 및 향후 연구 방향에 대해 서술한다.

제 2 장 Previous Research

2.1 Deep Generative Model

Generative Model은 특정 분포를 따르는 새로운 sample을 추출하는것을 가능하게 한다. 더 나아가, Deep generative Model 의 발달로 인해, 해당 분포의 explicit form을 추정하지 못하더라도, 새로운 sample 추출이 가능하다. 이러한 이유로, Deep Generative Model은 Super-Resolution, Text to speech, Drug discovery, Model based RL 등과 같은 여러 분야에서 널리 활용되고 있다. 이러한 Deep generative Model의 대표적인 모델은 Variational Autoencoder (VAE) 와 Generative Adversarial Nets (GAN)가 존재하며, 해당 모델의 다양한 변형들이 심도있게 연구되고 있다. 본 장에서는 본 논문의 기반이 되는 VAE 와 GAN, 그리고 VAEGAN 모델에 대해 서술한다. 본 논문에서 사용되는 변수와 그 정의는 Table 2.1에 제시하였다.

표 2.1: Table of Notation

Symbol	Description	Symbol	Description
N	# of data	K	# of latent feature
D	Discriminator	G	Generator
ϕ	Parameter of encoder (q)	θ	Parameter of decoder (p)
z^l	l_{th} sample of posterior distribution	L	# of sample of z^l
z_p	Sample from the $N(0, I)$ of posterior distribution	x	True data

2.1.1 Variational Auto-encoder; VAE

Kingma et al. [1] 은 reparametrization trick 을 variational lower bound 에 적용한 Stochastic Gradient Variational Bayes (SGVB) estimator 를 제안하였으며, 이를 활용하여 variational distribution 의 parameter 를 추정하는 Autoencoding Variational Bayes (AEVB) algorithm 을 제안하였다. AEVB의 두가지 구성요소인 recognition model $q_{\phi}(z|x)$ 과 generative model $p_{\theta(x|z)}$ 에 neural network를 활용하여 autoencoder 의 관점으로 해석한 VAE 또한 함께 제안하였다 [1]. 우리는 VAE 의 목적식과 Graphical Model 을 Eq. 2.1 과 Fig.2.1 에 각각 표현하였다. \tilde{L}_{VAE} 는 L_{VAE} 의 첫번째 항을 효율적으로 estimation 하기 위해, L 개의 sample $z^l(l = 1, \dots, L)$ 을 posterior distribution $q_{\phi}(z|y)$ 에서 뽑아 추정한 목적식 이며, z^l 에

reparametrization trick 을 적용하여, 효율적으로 backpropagation 되도록 변경할 수 있다.

$$L_{VAE} = E_{q_\phi}[\log p_\theta(x|z)] - KL[q_\phi(z|x)||p(z)] \quad (2.1)$$

$$\tilde{L}_{VAE} = \frac{1}{L} \sum_l^L E_{q_\phi}[\log p_\theta(x|z^{(l)})] - KL[q_\phi(z|x)||p(z)] \quad (2.2)$$

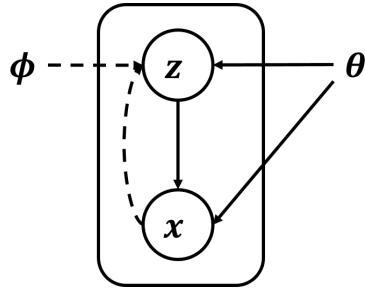


그림 2.1: Graphical model of VAE

2.1.2 Generative Adversarial Network; GAN

또 다른 대표적인 DGM 은 GAN 이다. GAN 은 VAE 와 동일하게, maximum likelihood estimation 을 통해 특정 분포를 추정한다는 점에서는 동일하지만, VAE 와는 다르게, 분포의 explicit form 을 가정하지 않고 분포를 추정할 수 있는 implicit density estimation model 이다. GAN 은 Generator 와 Discriminator 로 구성되며, Generator 와 Discriminator 가 Nash 균형을 이룰 때, 주어진 데이터의 분포를 asymptotically consistent 하게 추정할 수 있다고 보장되어 있다. GAN 모델은 VAE 와는 다르게, Pixelwise loss 보다는 Jensen shannon divergence 를 목적식으로 활용한 모델로, VAE 보다 비교적 선명한 이미지를 얻을 수 있다 는 장점이 있다. Fig. 2.2 와 Eq. 2.3 는 GAN의 구조와 목적식을 나타낸다. Discriminator D 의 parameter 는 목적식 L_{GAN} 을 maximize 하기 위해 학습이 된다. 반면, Generator G 에 속한 parameter는 L_{GAN} 을 minimize 하여, noise sample z_p 을 활용하여 생성한 $G(z_p)$ 를 Discriminator 가 제대로 구분하지 못하도록 학습된다.

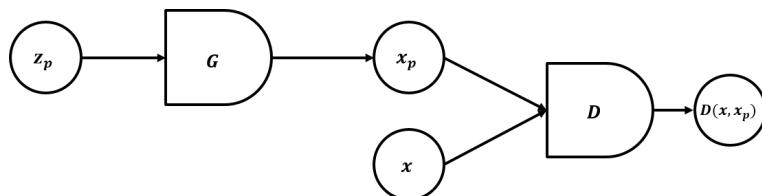


그림 2.2: Structure of GAN

$$L_{GAN} = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z_p \sim p_{z_p}(z_p)}[\log(1 - D(G(z)))] \quad (2.3)$$

2.1.3 VAEGAN

최근에는, 이러한 VAE 와 GAN 의 장점을 활용한 다양한 변형들이 연구되고 있다. 그 중 대표적인 연구는 Larsen et al. [3]이 제안한 VAE/GAN 모델이다. 기존의 VAE 목적식은 reconstruction error 와 prior 를 활용한 regularization term 으로 구성된다. 특히, reconstruction error 는 Mean squared error 와 같은 elementwise-metric 이 주로 활용된다. 하지만, 이러한 elementwise-metric은 이미지와 같은 task 에 적합하지 않다. 실제 사람이 이미지를 보고 판단할 때는, pixelwise한 특성보다 큰 개념에서 이미지를 바라보고 판단하기 때문이다. 이는 이미지 뿐만 아니라, Reinforcement learning 에서도 동일하게 적용된다. 두 policy network 가 도출하는 개별적인 action 들의 차이로 policy network 를 판단하기 보다는 전체적인 action 의 분포가 더 중요하기 때문이다. 그 까닭에 elementwise-metric의 단점을 보완할 수 있는 보다 다른 개념의 loss가 필요하며, Larsen et al. [3] 은 VAE와 GAN 을 결합하여 위의 문제를 해결하였다.

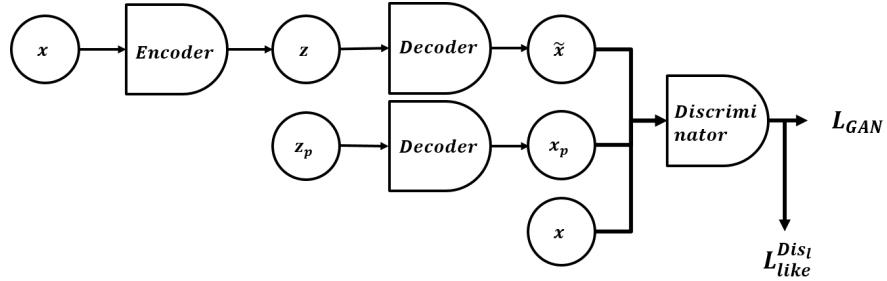


그림 2.3: Structure of VAE/GAN

Fig. 2.3 와 Eq. 2.4~2.7 는 VAE/GAN의 구조와 목적식을 나타낸다. VAE/GAN 의 목적식은 크게 3 가지로 구성된다. 첫번째는 prior 관련 loss이며, 이는 VAE에서의 식과 동일하다. 두번째 loss는 featurewise loss 이다. 기존의 VAE 의 reconstruction error 와 유사하지만, 최종 output 이 아닌, 중간 feature level 에서 reconstruction error 를 계산한 향이다. 이는, 기존의 elementwise 보다 높은 level 에서 error 를 판단하기 위함이다. 세번째는 GAN 에서의 Discriminator loss 이며, 이는 결국 Jensen shannon divergence 를

계산하여, 분포들간의 거리를 계산하는 항이다.

$$L_{prior} = D_{KL}(q(z|x)||p(z)) \quad (2.4)$$

$$L_{like}^{Dis} = -E_{q(z|x)}[\log p(Dis_l(x)|z)] \quad (2.5)$$

$$L_{GAN} = \log(Dis(x)) + \log(1 - Dis(Dec(z))) \quad (2.6)$$

$$L_{VAE/GAN} = L_{prior} + L_{like}^{Dis} + L_{VAE/GAN} \quad (2.7)$$

2.2 Conditional Deep Generative Model

또 다른 발전 방향은, Conditional DGM이다. 위에서 언급한 DGM의 경우에는 1) label 정보를 활용하기 어려우며, 2) data 생성시에 control 할 수 있는 요소가 존재하지 않는다. 물론, VAE의 경우에는 'random walk in the latent space'를 통해서, 연속성을 지닌 다양한 이미지를 생성할 수 있지만, 이 또한 explicit하게 control 할 수 있는 요소가 없다. 만약 label 정보를 효율적으로 활용할 수 있다면, 1) semi-supervised learning 문제에 적용할 수 있고, 2) data의 생성시에도 control 할 수 있는 요소를 추가할 수 있게 된다. Kingma et al. [4] 와 Osindero et al. [5] 은 conditional 정보를 추가하여, 위의 2가지 장점을 살린 Conditional VAE (CVAE) 와 conditional GAN (CGAN) 을 각각 제안하였다.

2.2.1 Conditional Variational Auto-encoder

CVAE는 기존의 VAE에 존재하는 recognition model ($q_{\phi(z|x)}$)과 generative model ($p_{\theta(x|z)}$) 부분에 label (c) 을 condition 으로 추가한 것으로, 추가적인 해당 condition 정보는 concatenation 을 통해 반영되고 있다. Fig. 2.4 와 Eq. 2.8 ~ 2.10 는 CVAE의 구조와 목적식을 나타낸다. Eq. 2.8 은 label인 c 가 존재하는 데이터에 대한 식이며, Eq. 2.9는 label이 존재하지 않는 데이터에 대한 식이다. 그리고 2.10는 위 두식을 결합한 전체 데이터에 대한 목적식이다.

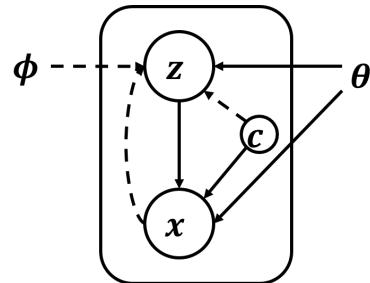


그림 2.4: Structure of CVAE

$$\log P_\theta(x, c) \geq E_{q_\phi(z|x, c)}[\log p_\theta(x|c, z) + \log p_\theta(c) + \log p(z) - \log q_\phi(z|x, c)] = -L(x, c) \quad (2.8)$$

$$\log P_\theta(x) \geq E_{q_\phi(c, z|x)}[\log p_\theta(x|c, z) + \log p_\theta(c) + \log p(z) - \log q_\phi(z, c|x)] = -u(x) \quad (2.9)$$

$$J = \sum_{(x, c) \sim p_l} L(x, c) + \sum_{x \sim p_u} u(x) \quad (2.10)$$

2.2.2 Conditional Generative Adversarial Network

CGAN 역시 CVAE와 유사하다. normal distribution 으로부터 생성된 z 와, label 정보인 c 를 함께 고려하여 Generator의 output 을 생성한 후, Discriminator 단계에서도 마찬가지로, label 정보인 c 를 고려하여 계산이 이뤄진다. CGAN 의 구체적인 Structure 와 목적식은 Fig. 2.5 와 Eq. 2.11 에 각각 제시하였다.

$$L_{CGAN} = E_{x \sim p_{data}(x)}[\log D(x|c)] + E_{z_p \sim p_{z_p}(z_p)}[\log(1 - D(G(z|c)))] \quad (2.11)$$

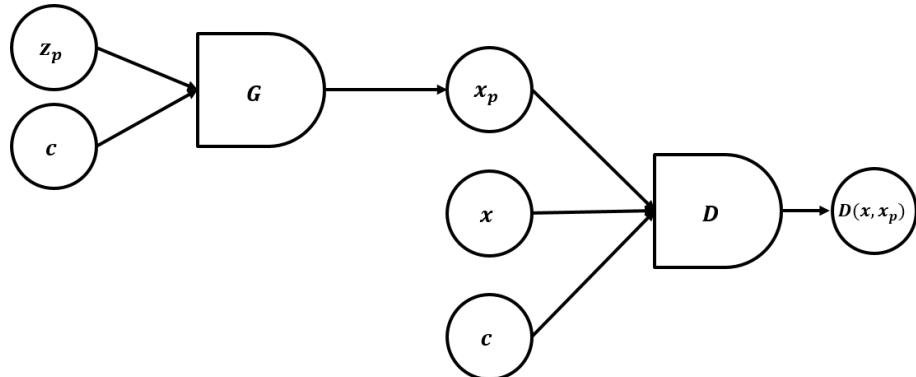


그림 2.5: Structure of CGAN

제 3 장 Methodology

본 논문에서 제안하는 모델 및 방법론에 대해 설명하고자 한다. 본 모델은 VAE-GAN의 구조를 차용하여 설계되었으며, 크게 3가지 부분으로 나눌 수 있다. 첫 번째는 encoder 부분으로, 주어진 input state data S 를 효율적으로 표현할 수 있는 z 로 mapping하게 된다. 두 번째 부분은 decoder로써, encoder를 통해 함축된 state S 에 대한 정보를 활용하여 적절한 action \tilde{A} 를 generate할 수 있다. 마지막으로 discriminator의 경우, input state S 에 해당되는 optimal policy A 와 decoder로부터 재생성된 \tilde{A} 를 input으로 받아 feature를 생성하고, A 와 \tilde{A} 를 구분하는 역할을 수행한다.

3.1 Network Overview

본 연구에서 새롭게 제안하는 모델은 크게 3가지 부분으로 구성되며, 전체적인 모델 구성은 그림 3.1에 제시되어 있다. 본 모델의 목적은 MDP로 모델링이 가능한 강화학습 문제에 대해 reward inference 없이, 주어진 state 정보에 대한 optimal action을 찾고자 함에 있다.

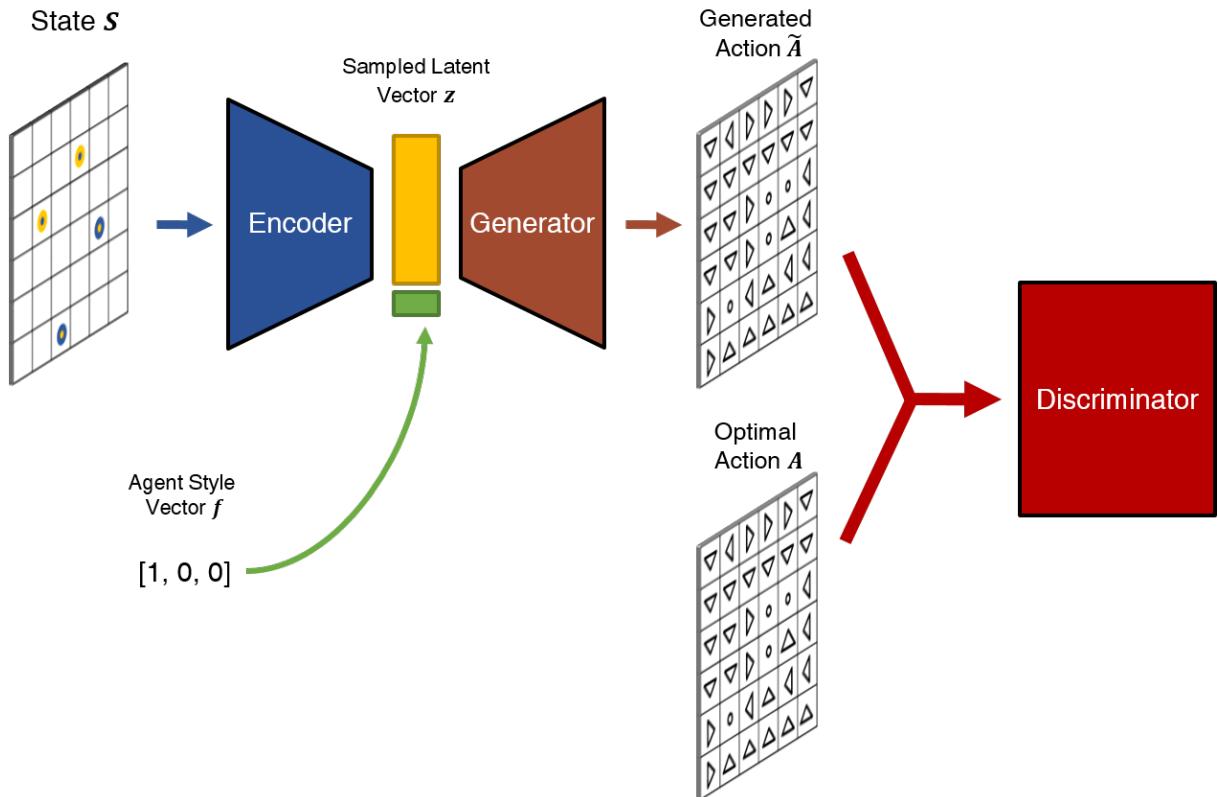


그림 3.1: Model Overview

그림 3.1에서 표시된 state S , agent style vector f 및 optimal action A 는 모두 dataset으로부터 주어진다. State S 는 $20 \times 20 \times 2$ 의 tensor data이며, agent style vector f 는 1×3 크기의 vector data이며, optimal action A 는 $20 \times 20 \times 1$ 크기의 matrix data로 주어지게 된다. Data로써 주어지는 S, A, f 의 의미에 대해서는 4.1장에서 자세히 다룰것이나, 모델의 역할에 대해 설명하기 위해 간략히 언급하자면 다음과 같다.

State S 는 grid-shape의 environment에서 agent의 행동을 유인하는 요소인 reward를 생성할 수 있는 object의 위치 및 특성 (inner & outer color)에 대한 정보를 담고 있다.

Encoder network를 통해 해당 state에 대한 정보는 latent vector z 로 encoding될 수 있다. Agent style vector f 의 경우, environment에 주어진 정보인 state에서 어떤 방식으로 행동할지를 결정하는 agent style에 대한 정보를 담고 있다. 해당 style vector f 를 state 정보 z 와 concatenate시킴으로써, 주어진 state에서 어떤 양식으로 행동할지에 대한 일종의 condition정보를 제공하는 역할을 한다.

Generator는 encoder를 통해 얻어진 state에 대한 정보 z 와, agent style에 대한 정보 f 가 concatenate된 vector를 input으로 받아, 주어진 state에 대해 제시된 agent style에 가미된 action 을 reconstruct 하는 역할을 수행한다.

Discriminator는 Generator로부터 생성된 action \tilde{A} 와, 사전에 주어진 optimal action A 를 input 으로 받으며, A, \tilde{A} 각각에 대한 feature 를 생성함과 동시에 real & fake data에 대한 판단을 확률값으로 생성하게 된다.

3.1.1 Encoder

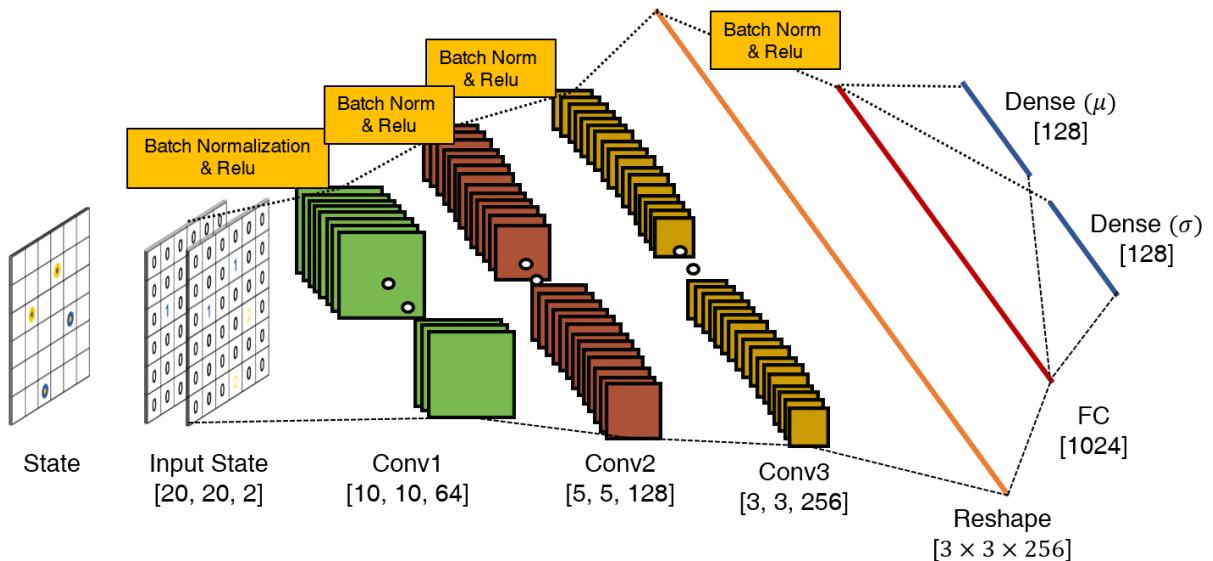


그림 3.2: Encoder Network

Encoder network의 구조는 그림 3.2으로 도식화 하였다. Model의 input으로 들어가는 state는 $[20 \times 20 \times 2]$ 의 크기를 가지는 tensor이며, convolutional layer를 거친 뒤, batch normalization을 진행하며, ReLU activation function을 통과하는 과정을 3번 거치게 된다. Convolution layer에서 사용되는 kernel은 $[5 \times 5 \times 2]$ 의 크기를 가지고, stride의 크기는 2로 설정하고 padding은 SAME 조건으로 설정하였다. 첫 번째, 두 번째 및 세 번째 convolution layer의 output dimension은 각각 64, 128, 256으로 설정하였고, 마지막 convolution layer를 통과한 뒤 reshape 과정을 거치고, fully connected layer의 dimension은 1028로 설정하였다. FC Layer의 결과에 대해 다시 한 번 output dimension을 128로 가지는 fully connected layer를 거치게 되어 μ 와 σ 를 얻고, 이를 encoder network의 output으로 산출하게 된다.

3.1.2 Generator

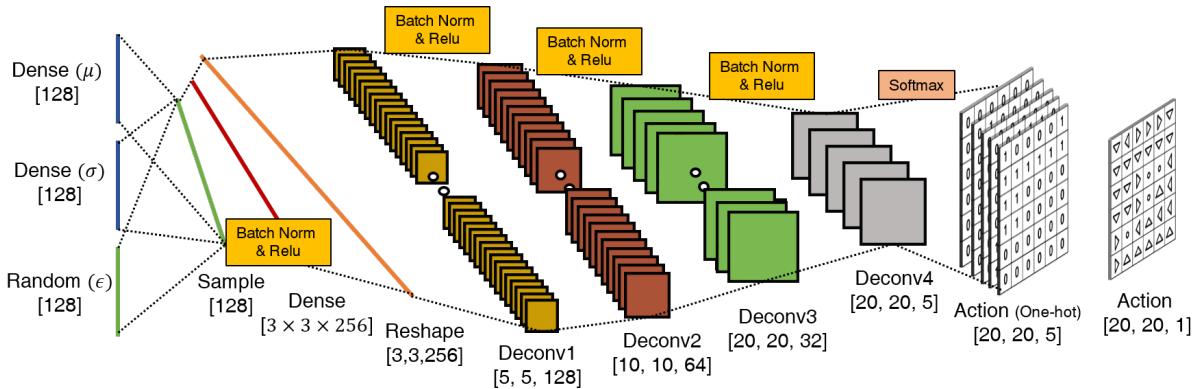


그림 3.3: Generator Network

Generator network의 구조는 그림 3.3로 도식화 하였다. Encoder를 통해 생성된 state에 대한 정보 code를 생성하기 위해, VAE에서 제시된 reparametrization trick을 활용하여 sampling된 vector code를 생성한다. 해당 code는 fully connected layer를 거쳐, batch normalization과 ReLU activation function을 통과한다. 이는 reshape 과정을 거쳐 $[3 \times 3 \times 256]$ 의 형태를 가지는 tensor로 변형된다. 해당 tensor는 4번의 deconvolution layer를 거치게 되며, 각각의 layer를 거치며 batch normalization과 ReLU activation function 을 통과한다.

VAE를 포함한 일반적인 Auto-Encoder 계열의 decoder(generator)는 encoder 단에서 주어진 데이터를 재생성하는데 그 목적을 가지나, 본 모델에서는 원래 주어진 state 정보 S 를 재생성하는데 목적이 있는 것인 아니라, 해당 state 및 agent style에 해당하는 optimal action을 복원하는 것을 목표로 한다.

이를 위해 VAEGAN에서 제시된 deconvolution 과정을 그대로 거치되, 마지막 네 번째 deconvolution 을 거친 이후 생성되는 tensor는 state가 아닌, 또 다른 데이터로 주어지는 action을 modeling하는 형태로

구성하였다. 해당 tensor는 softmax function을 통과함으로써 특정 cell에 대해 $\{0,1,2,3,4\}$ 의 5가지 action에 대한 확률 값을 가지도록 모델링 하였으며, 이렇게 재생성된 action \tilde{A} 는 $[20 \times 20 \times 5]$ 의 크기를 갖는 tensor가 된다.

3.1.3 Discriminator

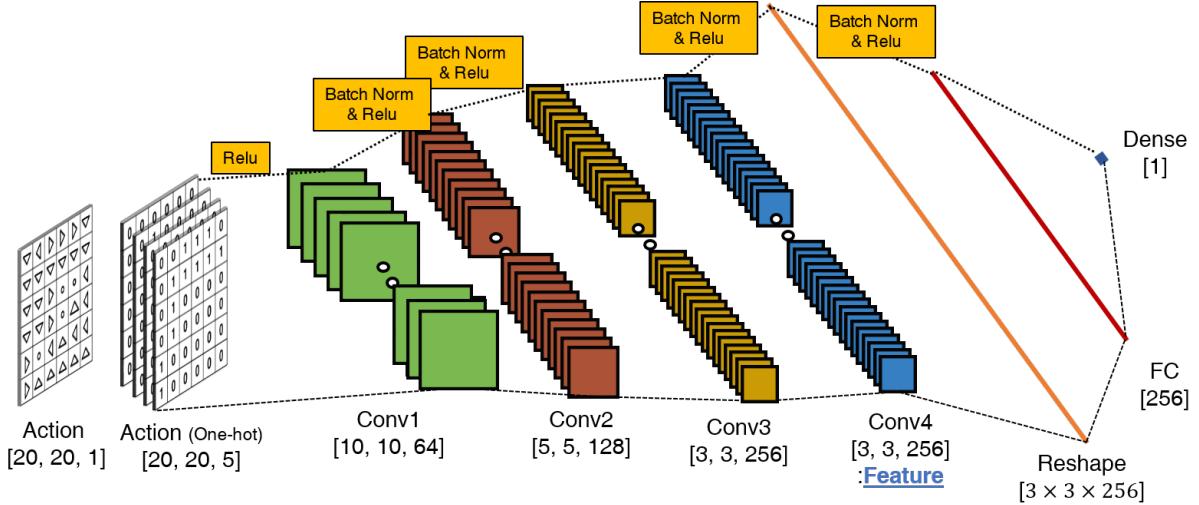


그림 3.4: Discriminator Network

Discriminator의 구조는 그림 3.4로 도식화 하였다. 전체적인 network의 구성은 encoder network와 유사한 구조를 가지고 있으나, discriminator의 output을 위해 약간의 변형된 구조를 가지게 된다.

먼저, discriminator network의 input은 generator로부터 생성된 tensor \tilde{A} 혹은 데이터로써 주어지는 optimal action A 가 된다. 해당 tensor는 각 cell에서 취할 수 있는 5가지 action 중 1개의 action을 one-hot encoding으로 표현한 $[20 \times 20 \times 5]$ 의 크기를 가지고 있다. 첫 번째 convolution layer에서 encoder network 와는 다르게 batch normalization을 거치지 않는 것이 특징이다. 또한 VAEGAN의 주요 특징인 feature를 산출하기 위해, encoder network의 convolution layer 구조보다 하나의 convolution layer가 추가된 형태를 가진다. 해당 layer를 거칠 때 또한 batch normalization을 수행하지 않고, activation function도 거치지 않은 convolution 연산만을 수행한다. 이렇게 생성된 $[3 \times 3 \times 256]$ 형태의 tensor를 input으로 주어진 action에 대한 feature로 return한다.

이후 reshape 과정 및 두 번의 fully connected layer를 거쳐 Discriminator loss 계산을 위한 output을 생성하게 된다.

제 4 장 Results

4.1 ObjectWorld Benchmark Dataset

본 연구에서 사용할 데이터를 확보하기 위해, Inverse Reinforcement Learning domain에서 활용되는 ObjectWorld Benchmark Model을 활용하였다. ObjectWorld Benchmark Model은 Reinforcement Learning domain에서 주요 예시로 활용되는 GridWorld Model을 확장한 것으로, 2011년 Levine[10]에 의해 처음으로 제안되었다.

4.1.1 MDP formalism

ObjectWorld Model은 Markov Decision Process (MDP)로 모델링 될 수 있는 요소들을 갖추고 있으며, 다음의 MDP로 정의될 수 있다.

$$\text{ObjectWorldMDP} = \{S, A, T, R, \gamma\}$$

$$S = \{\text{Cell}_{x,y}\}$$

$$A = \{\text{Up}, \text{Down}, \text{Left}, \text{Right}, \text{Stay}\}$$

$$T = P_a(s, s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$$

$$R = R_a(s, s')$$

$$\gamma = 0.9$$

ObjectWorld Model의 상태변수집합(state S)은 $N \times N$ 크기의 grid화되어 있는 environment상에서, agent가 현재 위치하고 있는 $\text{cell}_{x,y}$ 의 정보를 의미한다. 여기서 x, y 는 Grid 상의 agent 절대 위치를 의미한다. 행동변수집합 (action, A)은 Up, Down, Left, Right, Stay의 5가지 행동으로 이루어져 있으며, agent가 ObjectWorld environment인 grid cell에서 취할 수 있는 행동을 의미한다. Up, Down, Left, Right action의 경우 현재 agent의 위치에서 상대적인 위치 이동을 의미하는 것이며, Stay action의 경우 현재 위치를 다음 time step에서도 유지하는 action을 의미한다. 전이확률모델 (Transition Model T)의 경우, 현재 time step에서 agent의 state가 $s_t = s$ 로 주어지고, action $a_t = a$ 의 행위를 취하였을 때, 다음 time step에서 agent의 state가 $s_{t+1} = s'$ 될 확률을 의미한다. 전이확률모델은 $[|S| \times |A| \times |S|]$ 크기의 3차원 tensor로 표현이

가능하다. 보상함수 (Reward, R)의 경우, 현재 time step에서 agent의 state가 $s_t = s$ 로 주어지고, action $a_t = a$ 의 행위를 취하고, 다음 time step에서 agent의 state가 $s_{t+1} = s'$ 로 결정되었을 때 agent가 받게 되는 보상 r_{t+1} 을 의미한다.

4.1.2 Reward Setting: ObjectWorld

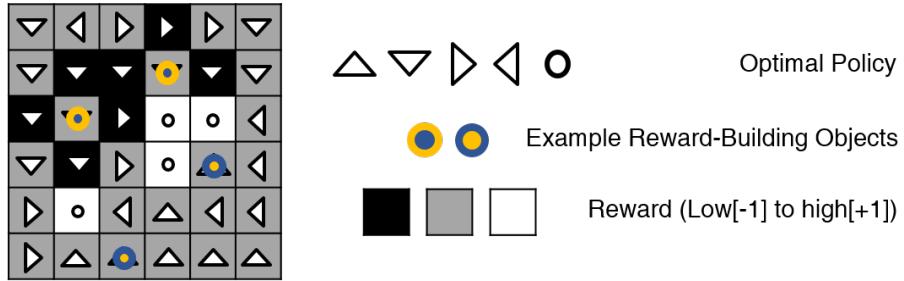


그림 4.1: Object World Benchmark Model: Concept

그림 4.1은 ObjectWorld Model의 구성 요소를 나타내고 있다. ObjectWorld Model은 grid-shape 형태의 environment 상에서, 임의의 위치에 reward를 생성하는 역할을 수행하는 reward shaping objects를 가진다. Reward shaping object는 2개의 다른 색 (inner color, outer color)을 내부 속성으로 가지며, 이 가운데 inner color는 distractor로의 역할만을 수행한다. 즉, 직접적으로 reward 구성에 영향을 끼치지는 않지만, agent가 인식할 수 있는 observation의 한 요소로 작용하게 된다. Reward를 구성하는 직접적인 요소는 object의 outer color로, 지정된 outer color로부터 얼마만큼의 상대적 거리(를 가지는지에 따라 해당 cell의 reward가 결정되게 된다. ObjectWorld에서의 거리는 Manhattan distance로 산출한다.

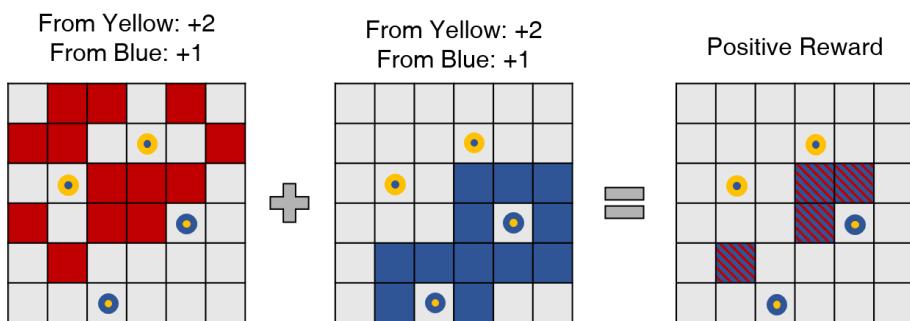


그림 4.2: Positive reward condition

그림 4.2는 agent에게 positive reward를 줄 수 있는 cell의 위치가 결정되는 과정을 그림으로 도식화 한 것이다. 예시로 주어진 ObjectWorld Model의 경우, environment의 크기가 6×6 으로 구성되어 있으며, 각각 $\{Yellow, Blue\}$ 를 inner, outer color로 가질 수 있는 4개의 object로 구성되어 있다. Reward의 경우 $\{-1:\text{Negative}, 0:\text{Neutral}, +1:\text{Positive}\}$ 의 값을 가질 수 있다.

Positive Reward는 두 개의 다른 색에 의해 결정된다. 즉, 서로 다른 outer color를 가지는 object로부터의 상대적 거리로 결정되는데, 그림 4.2에서 보여지는 예시의 경우에는 (Distance from yellow == 2) & (Distance from blue == 1)의 두 조건이 모두 만족하는 위치의 cell에 대해 +1의 reward가 형성되는 것을 확인할 수 있다.

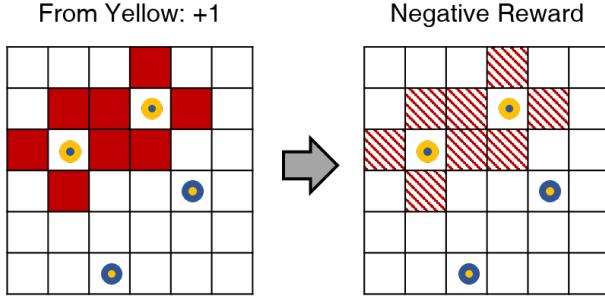


그림 4.3: Negative reward condition

반면 negative reward의 경우, 하나의 색에 의해서만 결정된다. 그림 4.3에서 보여지는 예시의 경우, (Distance from yellow == 1) 하나의 조건만을 만족하는 위치의 cell에 대해 -1의 reward가 형성되는 것을 확인할 수 있다.

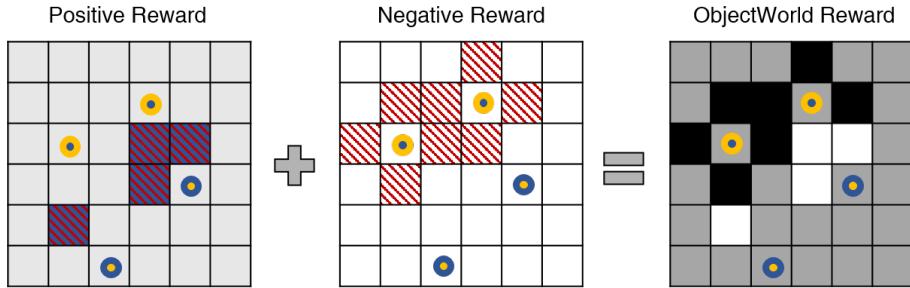


그림 4.4: Reward construction for given ObjectWorld

ObjectWorld model에서 positive, negative로 결정되지 않은 cell들에 대해서는 모두 Neutral한 reward 값인 0으로 설정되게 된다. 또, Positive reward와 Negative reward가 형성되는 두 조건을 모두 만족하는 cell의 위치에 대한 reward는 positive reward로 설정된다. 그림 4.4는 ObjectWorld에서 각 object의 위치에 따른 reward 형성 과정 전체를 개략적으로 표현하고 있다.

본 논문에서는 다른 style을 가지는 agent들이 같은 state를 마주하였을 때 행동하는 방식을 각 agent의 style에 맞는 optimal policy를 도출하는데 그 목적을 가진다. 이를 위해 다른 style의 행동 양식을 가지는 agent를 구현하기 위해, reward의 생성 조건을 다르게 하여 데이터를 산출하였다.

그림 4.5는 object의 위치가 동일하게 주어졌을 때, reward 생성 조건을 다르게 모델링 함으로써 각 style별로 상이한 reward 분포가 생성되는 것을 도식화 한 것이다.

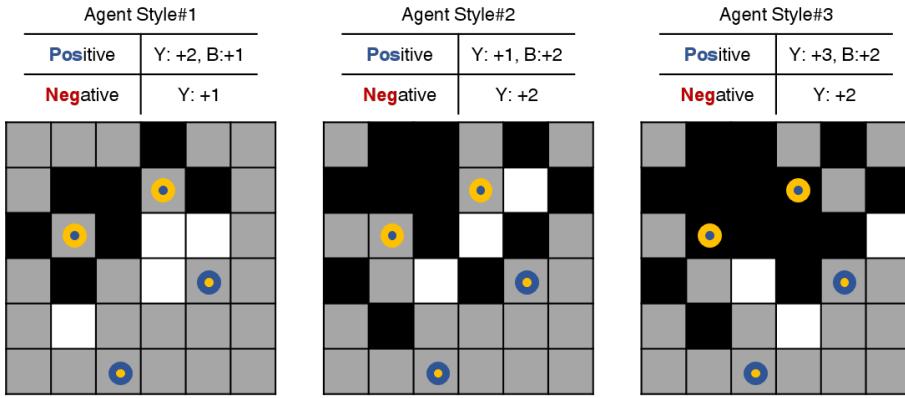


그림 4.5: Different reward generation from different agent style

4.1.3 Policy generation

본 장에서는 이전 장인 4.1.2절에서 생성된 reward에 대해, 본 논문에서 제안하는 모델의 training data로써 활용되는 optimal policy를 생성하는 과정을 설명한다.

생성된 각 cell별 reward에 해당하는 optimal policy를 찾기 위해, dynamic programming 방법론 중 하나인 value iteration을 사용하였다. Value iteration은 optimal value function이 존재할 때, 해당 값으로 수렴한다는 것이 알려져 있으며[11], value iteration을 통해 산출된 optimal value의 근사치인 $v(s')$ 과 해당 step에서의 reward $r(s, a, s')$ 를 활용하여 최대의 value를 산출할 수 있는 action을 선택하는 policy function을 return하도록 한다. Algorithm 1은 value iteration의 과정을 의사코드로 나타낸 것이다.

Algorithm 1 Value Iteration

```

repeat
    Initialize array  $v$  arbitrarily
    for each  $s \in S$  do
         $temp \leftarrow v(s)$ 
         $v(s) \leftarrow \max_a \sum_{s'} p(s'|s, a)[r(s, a, s') + \gamma v(s')]$ 
         $\Delta \leftarrow \max(\Delta, |temp - v(s)|)$ 
    end for
    until  $\Delta < \theta$  (a small positive number)
    return  $\pi(s) = \arg \max_a \sum_{s'} p(s'|s, a)[r(s, a, s') + \gamma v(s')]$ 

```

4.1.4 Generate Dataset for the Network Training & Testing

본 절에서는 본 논문에서 제안하는 network를 학습시키는데 필요한 데이터들을 어떻게 구성하고, 생성하였는지 설명할 것이다. 앞서 설명하였던 ObjectWorld의 MDP 구성요소인 S, A, T, R, γ 가운데, 실험에서 사용하게 될 데이터는 T ; transition probability, γ 두 가지를 제외한 S, A, R 의 state, action, reward와, 마

지막으로 어떤 style의 agent로부터 생성된 데이터인지를 나타내는 style feature F 로 구성된다.

State의 경우, ObjectWorld의 environment를 구성하는 모든 cell의 정보를 담은 하나의 tensor를 생성한다. 이는 GAN을 비롯한 일반적인 Neural network를 학습시키는데 사용되는 이미지 형태의 tensor로 구성할 수 있다.

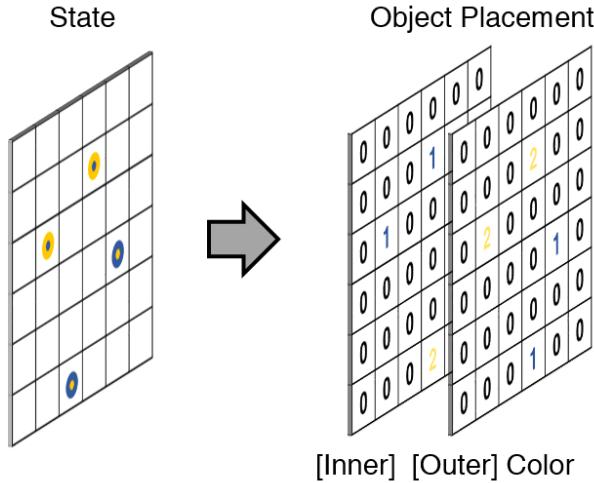


그림 4.6: Data generation of state

그림 4.6은 State 정보가 tensor로 변형된 형태를 도식화한 것이다. ObjectWorld의 environment size 가 N 으로 주어졌을 때, 변형된 tensor는 $[N \times N \times 2]$ 의 크기를 가진다. 생성된 Tensor의 마지막 dimension 은 2로 주어지는데, 이는 ObjectWorld에 존재하는 Object의 위치 및 inner/outer color에 대한 정보를 주 기 위함이다. Object가 존재하지 않는 cell의 경우에는 0의 값을 가지는데, 이는 agent가 인식할 수 있는 (Observable) 정보만을 제공하기 위함이다. ObjectWorld에서의 agent는 object의 위치 및 색의 정보만을 인식하고, 이들이 생성하는 reward에 대한 정보는 주어지지 않는다.

State를 제외한 나머지 요소인 A, R, F 의 경우, 1개 state당 agent style 개수 만큼 대응대는 수의 데이터 가 생성된다. 본 논문에서 설정한 agent style의 수는 3개로, 1개 state에 대해 3종류의 style을 가지는 agent 가 있다고 가정하였다.

지난 4.1.2절에서 설명한 바와 같이, agent style의 수 만큼 다른 reward 분포가 생성되게 되며, 다르게 설정된 reward의 경우 4.1.3 절에서 설명한 value iteration 과정을 통해 다른 policy function을 가지게 된다. Style과 reward, policy는 각각 1:1 대응관계를 가진다.

그림 4.7는 reward, style, action 데이터가 생성되는 방식에 대해 도식화 한 것이다. Reward의 경우, 각 cell에 위치하게 되었을 때 얻게되는 reward $r_{t+1} = r(s_t, a_t, s_{t+1})$ 에 대한 정보를 담고 있으며, $[N \times N \times 1]$ 의 크기를 가진다. Style의 경우, agent 중 어떤 style을 가지는 agent에 대한 reward 정보인지를 one-hot



그림 4.7: Data generation of reward, style, and action

encoding을 통해 vector로써 표현한 것이며, 데이터의 크기는 $[1 \times 3]$ 이다. Action의 경우, 각 cell에 위치하게 될 때 어느 행동을 취할 것인지에 대한 정보를 담고 있으며, 각 방향별 이동 및 stay 총 5개의 action을 encoding하기 위해 정수 $\{0,1,2,3,4\}$ 의 표현 방식을 활용하였다.

표 4.1: ObjectWorld Model Parameters

ObjectWorld Model Parameters	Value
Environment Size	20
# of object	15
# of color	2
# of unique state	40,000
# of style	3
Positive Reward condition	1:{3,2}, 2:{1,2}, 3:{2,3}
Negative Reward condition	1:{2}, 2:{2}, 3:{3}
γ :Discount factor	0.9
θ :Optimal value threshold	0.01

표 4.1은 실제 학습에 사용하기 위해 생성한 ObjectWorld Model의 parameter을 정리한 것이다. Unique state의 경우, $[20 \times 20]$ 크기의 environment 상에서 object의 위치 및 색이 유일하게 존재하는 경우들만을 고려한 데이터의 크기이다. γ 와 θ 의 경우, value iteration을 통해 optimal action을 뽑아낼 때 사용되는 parameter로 설정되는 값이다.

표 4.2: Data Statistics

Data class	Value	Data class	Value	Data class	Value	Data class	Value
Training	102,000	Style 1 train	36,000	Style 2 train	36,000	Style 3 train	36,000
Testing	18,000	Style 1 test	4000	Style 2 test	4000	Style 3 test	4000

표 4.2는 ObjectWorld Model로부터 생성된 데이터의 학습 및 validation에 사용되는 training/testing data의 분포를 나타낸 것이다. ObjectWorld의 unique state가 40,000개로 설정되었고, 3개 style의 agent를 학습시키기 위해 총 120,000개의 데이터를 생성하였다. 각 style별 데이터의 수는 동일한 비율로 산출하였으며, training과 testing data의 비율은 9:1로 구성하였다.

4.2 Experimental Setting

3장에서 설명한 전체 모델의 경우, agent style vector를 사용하여 condition을 주어 학습시킴으로써 각 style별 state에 대응되는 action을 얻는데 그 목적이 있다. 본 절에서는 전체 모델에 대한 실험을 진행하기 이전에, 1개 agent style에 대해 naive VAEGAN model을 구현하고, training을 진행하였다. 이는 본 논문에서 제안하는 모델로 발전시키기 이전에, conditioning을 하지 않은 latent code에 대해 action이 잘 생성되는지 확인하기 위함이다. 표 4.3은 해당 naive VAEGAN을 학습시킬 때 사용한 parameter를 정리한 것이다.

표 4.3: Model Parameter

Parameter	Value
Batch size	32
Max Iteration	270,000
Latent code dimension	128
Initial learning rate	0.0003
Learning epoch	150

4.3 Quantitative Results

Naive VAEGAN을 유일한 agent style action에 대해 학습 시킬 때, generator network에 의해 생성된 generated action \tilde{A} 가 유의미하게 학습되었는지 정량적으로 확인할 수 있어야 한다. 제안하는 정량적 measure는 생성된 action을 일정 step 수행하였을 때 계산된 cumulative reward와 optimal action을 수행하였을 때 계산된 cumulative reward를 비교하는 것이다. \tilde{A} 가 optimal action의 분포에 맞게 적절히 생성되도록 학습된다면, training iteration이 진행될수록 optimal action을 수행했을 때의 cumulative reward로 수렴할 것으로 기대할 수 있다.

그림 4.7은 training이 진행되는 동안 생성된 generated action \tilde{A} 와 optimal action A 에 대해 policy propagation을 진행하고, 그동안 만들어진 궤적 (trajectory)의 위치에 해당하는 true reward 값을 누적하

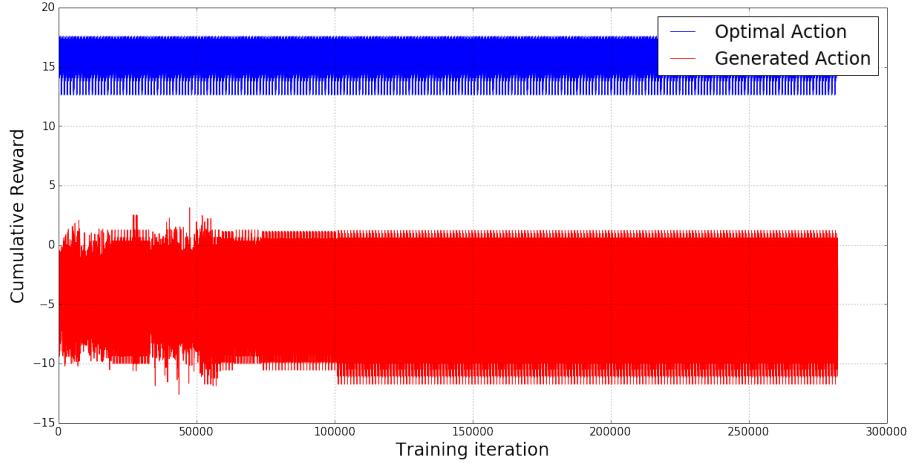


그림 4.8: Cumulative reward during 20 policy propagation for generated and optimal action

여 산출한 것이다. Model training 과정에서의 데이터는 32개의 batch를 유지하며 입력되기 때문에, 본 그래프에 plotting 된 값은 32개 cumulative reward의 평균값을 취하였다.

Graph의 결과를 보면, generated action과 optimal action 각각에 대한 policy propagation을 20 step 진행하였을 때, 누적된 reward의 변화를 관찰할 수 있다. 두 그래프 간 벌어져 있던 간극이 학습 epoch 이 진행되면서도 줄어들지 않는 것으로 보아, generator를 통해 생성된 action \tilde{A} 가 optimal action A 를 성공적으로 모방하지 못하고 있다는 것을 유추할 수 있다.

4.4 Qualitative Results

VAEGAN의 Training이 진행되는 과정에서 Qualitative Results를 평가하기 위해, training^o이 진행되는 epoch^o에 따라 변화하는 generated action과 동일한 state에서의 optimal action을 비교하고자 하였다. 그림 4.9는 training^o이 진행될 때, 동일한 state가 주어지는 상황에서 생성된 generated action과 optimal action을 color-coding으로 표현한 것이다.

그림 4.9의 가장 윗 부분은 training epoch^o 10일 때, 동일한 state 상황에서 생성된 generated action 은 왼쪽에, optimal action은 오른쪽에 도식화하였다. Colorbar가 나타내는 0,1,2,3,4의 숫자는 각각 해당 cell에서 취하는 action을 의미하며, 순서대로 {Down, Right, Up, Left, Stay} action을 의미한다. Training 초기 생성된 action^o \tilde{A} optimal action과는 전혀 다른 action을 생성하는 것을 확인 할 수 있다. Training 이 어느 정도 진행된 Epoch 50, 100의 경우를 살펴보았을 때도 \tilde{A} 가 optimal action A 를 정확히 모방하고 있지 못하는 것을 확인 할 수 있다.

특기할 점은, \tilde{A} 가 A 만이 가지는 특성을 가지고 있는 점이다. Positive reward를 가지는 cell에 근접한 cell들이 positive reward를 가지는 cell에 도달 할 수 있는 action을 취하는 경향을 확인할 수 있다.

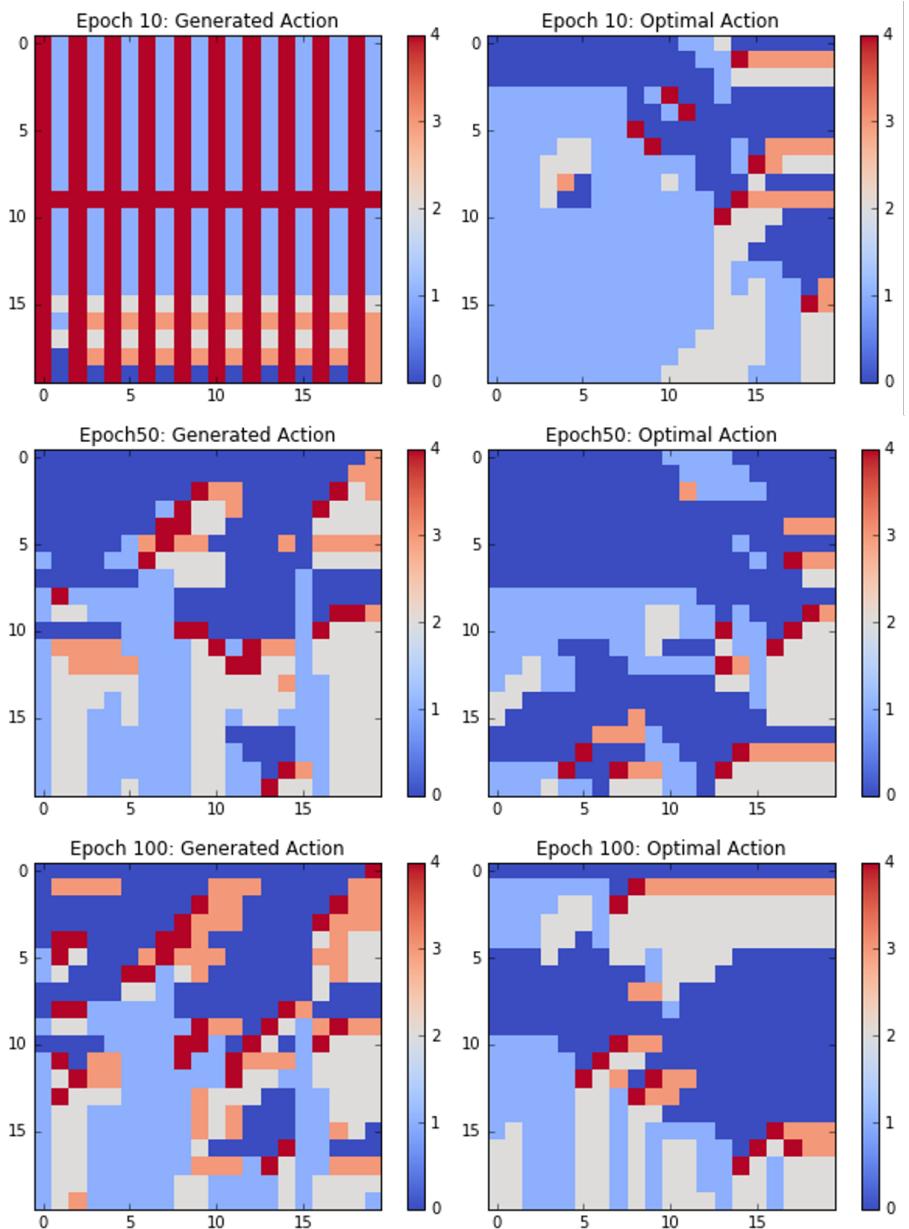


그림 4.9: Generated action & Optimal Policy for same state. (Top: Epoch 10, Middle: Epoch 50, Bottom: Epoch 100)

제 5 장 Discussion

본 장에서는 4.3절, 4.4절에서 살펴본 바와 같이, ObjectWorld Model에서 생성된 state, action data의 학습에 성공하지 못한 원인에 대해 추론하고, 해당 원인에 대한 개선 방법론들을 제시한다. 또한 개선 방향으로 제안한 방법론에 대해 설명하고, 해당 실험 결과 또한 서술하였다. 원인을 찾기 위한 대분류로는 두 가지를 설정하였다. 첫째로 ObjectWorld Model에서의 Dataset 관련 요인, 둘째로는 Network structure 및 parameter로 인한 원인으로 나누어 서술한다.

5.1 Improvement Direction

5.1.1 ObjectWorld Dataset

Size of dataset

표 4.2에서 언급한 바와 같이, 실험을 위해 생성된 ObjectWorld data는 unique state를 40,000개로 설정하였다. Agent style을 3개로 설정하였기 때문에, 1개 state당 3개의 style vector, reward, optimal action의 데이터가 생성되게 된다. VAEGAN의 training에서 사용된 데이터는 unique state가 36,000개로 구성되어 있다.

Environment size가 20으로 설정되고, 15개의 object를 가지며, 각각의 object가 inner color, outer color로 가질 수 있는 색의 선택지가 2개로 설정되어 있으므로, 현재 설정된 ObjectWorld Model의 경우, state의 전체 가능한 경우의 수는 아래와 같다.

$$\text{Object Position: } 400 \times \dots \times 386$$

$$\text{Object Color: } (2^2)^4$$

$$\text{Total number of possible state: } 1.6113e + 29$$

반면, training으로 제공한 36,000개의 unique state data는 전체 가능한 state의 경우의 수 가운데 2.2343e-23%만을 차지할 뿐이다. 물론 Deep Generative Model들의 학습 데이터로 사용되는 benchmark image dataset들의 경우에는 image를 이루는 각 pixel의 값을 input으로 받기에, unique state의 정량적인 비율 비교는 큰 의미를 가지지 못할 수 있다. 그러나, ObjectWorld Model은 각각의 unique state가 모두

지정된 reward로 mapping이 가능한, 다시말해 유의미한 데이터를 가진다. Negative Reward 형성 조건의 경우, 다른 object의 위치와 상관없이 negative reward에 대응되는 색을 outer color로 가지는 object가 존재한다면 -1의 reward를 형성하게 되고, 이는 value iteration을 통해 environment의 전체 cell에서의 optimal action 선택에 영향을 미치게 된다.

따라서 ObjectWorld environment의 크기를 좀 더 줄이고, 위치하는 object의 수를 줄이면서도 유의미한 reward가 형성될 수 있도록 dataset의 parameter를 조정함으로써 유의미한 학습을 시킬 수 있을 것으로 예상된다.

Element of State Data

VAEGAN에서 encoder network의 input으로 들어가는 state data의 경우, $[20 \times 20 \times 2]$ 크기의 tensor 중 단 30개의 cell만이 해당 cell에 위치하는 object의 color 특성을 표현하기 위한 0이 아닌 값을 가지며, 나머지 cell은 모두 0의 값을 가지고 있다. Convolution 연산에서 사용되는 kernel은 grid-shape data에 대해 인접한 grid의 값에서 feature를 뽑아내는 것을 그 목적으로 하고 있으나, 약 4%를 제외하고 나머지 cell이 모두 0의 값을 가지는 tensor를 사용해서는 input state의 유의미한 feature를 encoding하는 것이 쉽지 않을 것으로 예상된다.

이를 보완하고자, 단순히 ObjectWorld의 environment의 각 cell에 위치하는 object의 위치 정보 및 color 정보만을 state로 modeling하는 것이 아닌, 추가적인 feature를 state data에 추가하는 것이 학습에 도움을 줄 수 있을 것으로 예상된다.

3	2	2	1	2	3
2	1	1	●	1	3
1	●	1	1	1	2
2	1	2	1	●	1
3	2	1	2	1	2
2	1	●	1	2	3

그림 5.1: Distance from nearest object

예를 들어, 그림 5.1의 경우 현재 ObjectWorld의 environment에 존재하는 각 object의 위치가 주어져 있을 때, 가장 가까운 object에 대해 주변 cell이 가지는 manhattan distance를 표기한 것이다. 이는 가장 단순한 형태의 feature 형태이며, 지정된 innerouter color에 대해 가장 가까운 object로부터의 distance와 같은 형태로 생성이 가능하며, kernel weight의 학습에 유의미한 영향을 끼칠 수 있을 것으로 예상된다.

5.1.2 Network Structure & Parameters

Convolution Kernel

본 모델에서 convolution 연산 및 deconvolution 연산에 사용되는 kernel은 Channel dimension을 제외하고 $[5 \times 5]$ 의 크기를 갖는다. 반면 ObjectWorld에서 각 object의 위치에서 positive reward 혹은 negative reward의 영향을 받는 Manhattan distance는 최대 3을 넘지 않는다. 이에, 고정된 크기의 convolution filter를 사용하여 convolution 및 deconvolution 연산을 수행하는 것 보다, 다양한 크기의 filter를 적용하여 실험을 진행하여야 한다.

5.2 Experiment for Improvement

5.2.1 State Input Information

5.1.1절에서 제안한 개선방안 중, network 학습에 가장 큰 영향을 줄 것으로 생각되는 ObjectWorld dataset의 input 정보 변경안 (element of state data)에 대한 추가적인 실험을 진행하였다. State input data의 개선 방안은 총 두 가지로 제안하였으며 각 case는 아래 형식으로 나열하였다.

- **Manhattan distance** between Objects and each cell contained in ObjectWorld environment.
 1. Distance full feature for object: inner & outer color feature
 2. Distance part feature for object: Only outer color feature

기존 state 정보는 그림 4.6에서 도식화된 바와 같이 ObjectWorld 내부 object의 위치와 outer/inner color의 정보만을 표시하였기에, tensor내 non-zero entity가 sparse한 형태를 가지고 있어 pattern을 유추하기 힘들다는 단점을 가진다. 변경된 state data는 distance 기반의 정보를 담고 있으며, environment 내 정확한 object의 위치, 색 정보를 제공하는 대신, environment를 구성하는 각 cell에 대해 가장 가까운 Object에 대한 위치정보를 제공하였다. 이는 모든 cell위치에 대해 Object의 색, 위치 정보를 제공함과 동시에 input state tensor의 non-zero entity 정보를 dense하게 제공 할 수 있다는 장점이 있다. 기존 state 정보에서 본 state 정보로 바꾼 의도는, environment 내 object의 위치 정보를 모든 cell의 위치에 대해 dense하게 제공한다는 목적으로 이루어진다.

ObjectWorld 내 agent의 action 결정에 영향을 끼치는 것은 reward이며, reward 형성에 직접적인 영향을 제공하는 것은 각 object의 outer color이다. Inner color는 distractor로 reward 형성에 직접적으로 영향을 끼치지 않는 feature이기 때문에, inner/outer color 정보를 모두 제공하는 state와 outer color만을 제공하는 state 정보에 차이를 두는 것은 encoder가 action 형성에 영향을 끼치는 feature에 대해 선택적으로

latent code를 생성하고, 해당 정보를 통해 generator가 action을 효율적으로 생성할 수 있는지를 구분하기 위한 input data의 분류이다.

State 정보를 변경하여, 추가적인 실험을 진행하였다. 기존 VAEGAN에서 수행한 network 구조를 그대로 유지하되, input으로 들어오는 state 정보만을 변경하여 추가 실험한 결과를 도식화 하였다. 학습은 90,000 training epoch까지 진행하였으며, 그림 5.2, 5.3는 VAEGAN network를 학습시킬 때, loss 변화를 graph로 나타낸 것이다. 그래프로 나타낸 loss는 총 6가지로, 각각 1) Discriminator loss, 2) Generator loss, 3) Encoder loss, 4) Log-Likelihood loss, 5) KL-divergence loss, 6) Reconstruction loss (element-wise difference)로 나뉘다.

1. **Discriminator loss:** Discriminator loss for (Fake data + True data + Generated data)
2. **Generator loss:** Generator loss for (Generated data + True data) - Scaled log-likelihood loss
3. **Encoder loss:** KL-divergence loss - Log-Likelihood loss
4. **Log-Likelihood loss:** Negative log-likelihood feature loss between true data and generated data
5. **KL-divergence loss:** Prior loss
6. **Reconstruction loss:** Element-wise difference between true data and generated data

그림 5.2는 Manhattan distance를 기준으로, inner color 및 outer color 모두에 대한 feature를 input으로 network를 학습시켰을 때 loss 변화를 graph로 나타낸 것며, 그림 5.3는 outer color만을 feature input으로 하여 network를 학습시키는 상황에서 loss 변화를 도식화 한 것이다.

그림 5.2과 5.3 loss에 대한 의미 분석을 진행하였다. Discriminator loss를 보면, full feature의 경우 초기 30,000 epoch까지, part feature의 경우 초기 24,000 epoch까지의 학습을 통해 1차적으로 수렴한다. 이후 몇 번의 peak를 거치며 generator를 통해 생성된 data와 true data distribution에서 sample된 data간의 구별을 학습하는 것으로 보여지며, 이는 지속적으로 낮은 loss를 유지하면서 수렴하는 것을 통해 확인 할 수 있다. Generator 및 encoder loss의 경우, 두 feature 모두 discriminator의 학습이 진행됨에 따라 변동되는 것을 확인할 수 있다. 일반적인 GAN의 학습에 있어 해당 network가 적절하게 학습되었는지를 확인하는 방법은, 첫 번째로 reconstructed image의 완성도를 정성적으로 확인하는 방법이며, 두 번째는 generator loss 가 줄어들며, 수렴하는지를 확인하는 것이다. Full feature를 input으로 가졌을 때 및 part feature를 input으로 가지는 두 가지 경우에 대해 90,000 epoch까지 학습이 진행된 결과를 확인하였다. 두 경우 모두 30,000

epoch 초반, discriminator loss가 급격한 peak를 가질 때 generator loss가 감소하지만, 이후 지속적으로 증가하는 것을 볼 때 generator의 학습이 정상적으로 진행되고 있지 못하고 있다는 점을 알 수 있다.

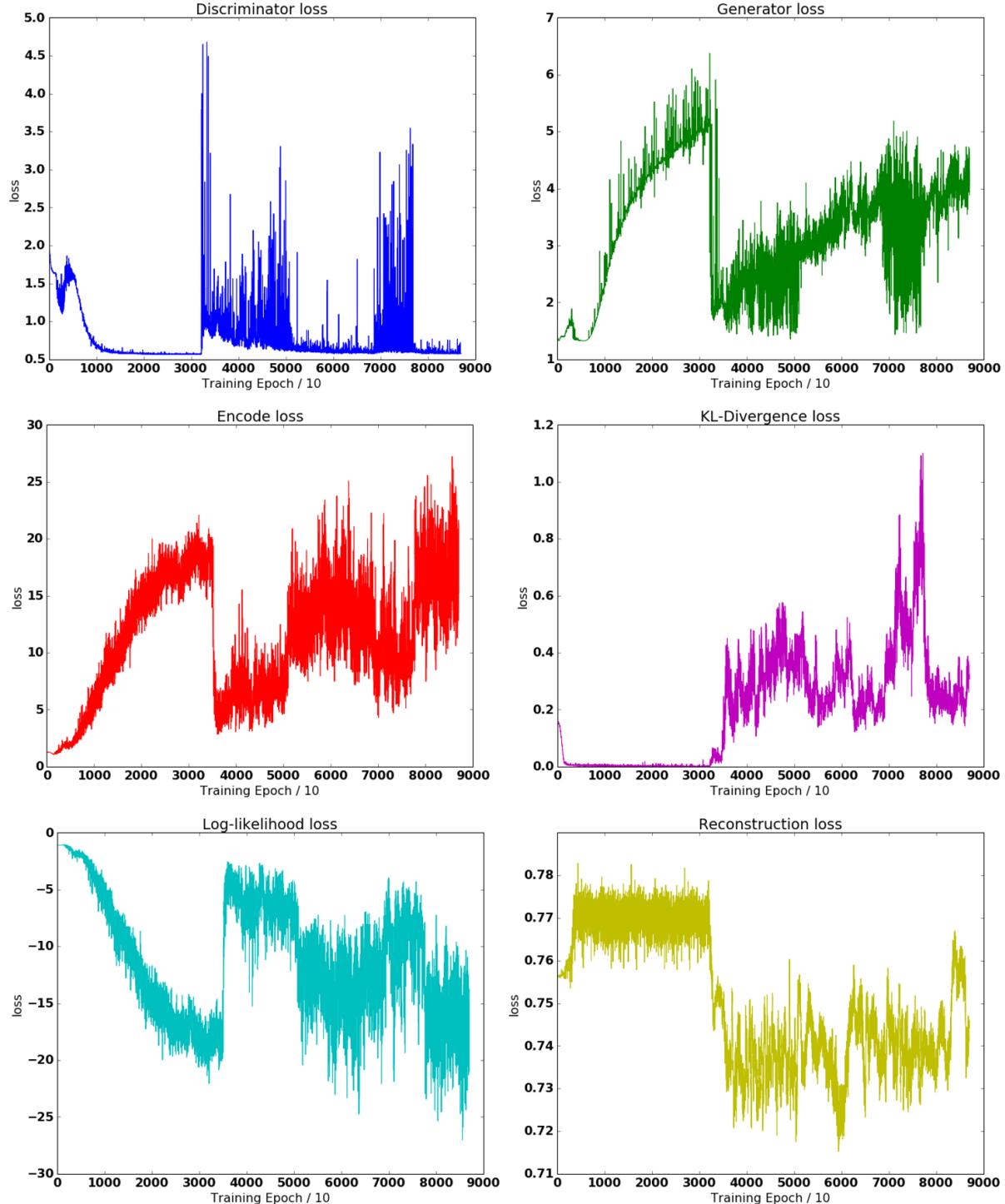


그림 5.2: Experiment loss result: Full(Inner/outer) feature

마지막으로 reconstruction loss는 generated data와 true data의 element-wise difference term으로 이루어져 있고, 낮아질수록 이상적으로 학습된다고 볼 수 있다. Full feature input의 경우 35,000 epoch 이후

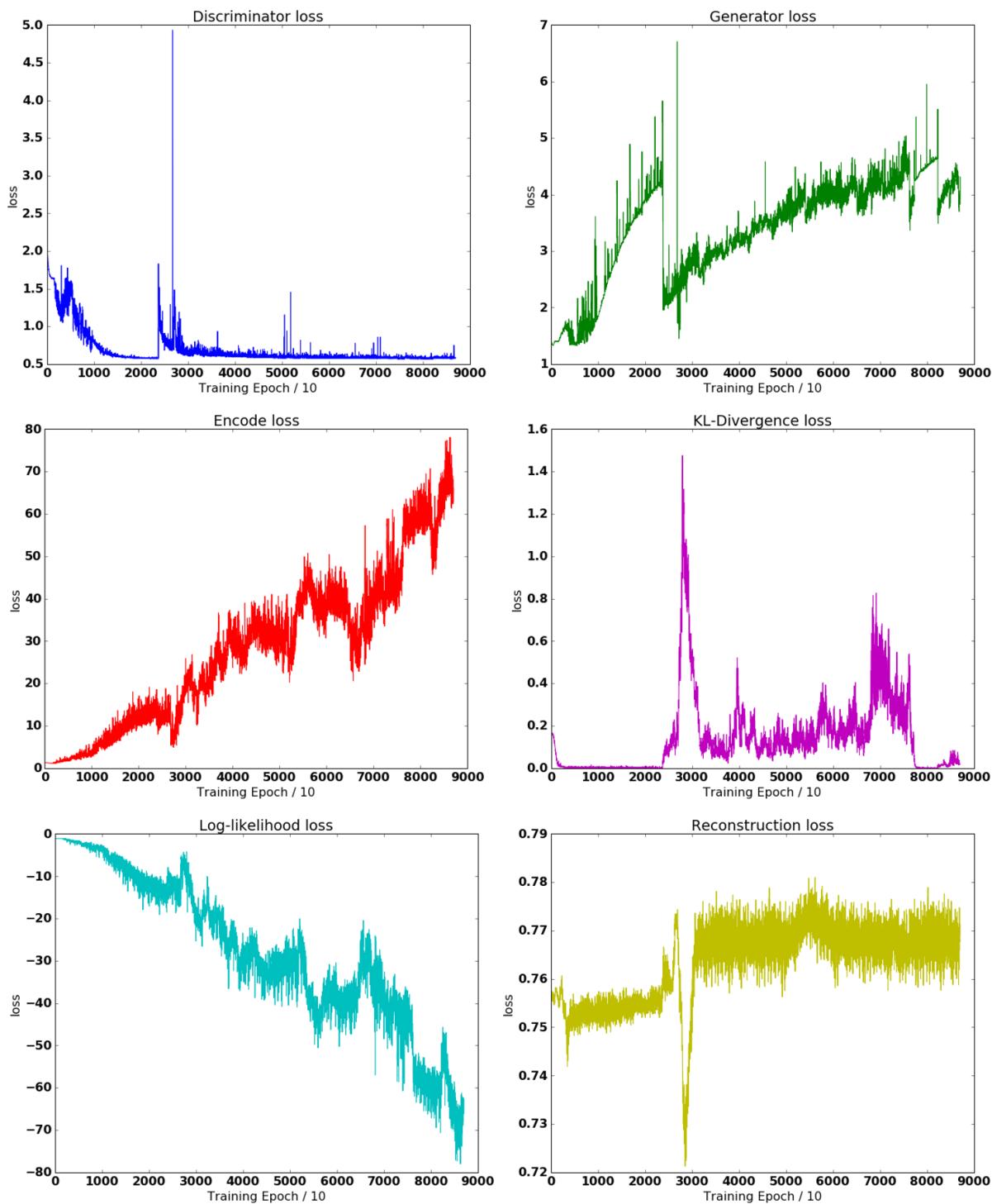


그림 5.3: Experiment loss result: Part(outer) feature

reconstruction loss가 일정 감소하나, 다시 증가하는 추세를 확인할 수 있으며, part feature의 경우 30,000 epoch때 현저히 감소하는 결과를 보이나, 이후 지속적으로 loss가 증가하는 경향을 보이는 것을 확인할 수 있다.

제 6 장 CONCLUSION AND FUTURE WORK

본 논문에서 제안하는 모델은 3장에서 언급한 바와 같이, 제한된 state에 대한 정보가 주어질 때, training 과정을 거쳐 encoder network가 state의 특정 위치마다 존재하는 object의 위치 정보 및 특성과 주변 cell 간의 관계를 잘 encode 할 수 있도록 학습되어야 한다. 또한 generator network는 encoder network로부터 생성된 latent code vector와 style vector가 concatenate된 정보를 통해 주어진 input state에 대해 지정된 style을 반영하여 각 cell에서 어떻게 행동해야 하는지에 대한 action \tilde{A} 를 생성할 수 있어야 한다. 마지막으로 discriminate network는 생성된 action \tilde{A} 와 optimal action A 각각의 feature를 비교하고, $Dis(Gen(z))$ 와 $Dis(x)$ 의 차를 최소화할 수 있도록 학습하여 optimal action에 가까운 generated action \tilde{A} 를 생성할 수 있도록 학습하여야 한다.

동일한 state가 주어졌을 때, 서로 다른 행동을 취해야 하는 heterogeneous agent가 존재하는 상황을 생각해 볼 수 있다. 예를 들어, 토목 공정을 수행하는데 필요한 기기는 굴삭기와 트럭을 예로 들 수 있는데, 이들은 토목 공정을 진행함에 따라 절도되고 축적되는 토양의 분포를 관찰할 수 있다. 이기종의 두 기기가 받아들이게 되는 토양의 분포는 같으나, 엄연히 다른 task를 수행해야 하는 입장에서 둘의 style이 다르다 표현할 수 있다. 이들의 다른 style은 동일한 토양 분포에 대해서도 특정 지점에 대한 reward를 다르게 가질 수 있으며, 행동 양식 또한 다르다. 본 모델은 위 예시와 같이 이기종의 개체가 존재하는 상황에 대해 최적의 행동을 수행할 수 있도록 제안되었다.

향후 연구 개선 방향으로는 5절에서 언급한 대로 training dataset의 변형이 필요하다. 학습이 가능하도록 ObjectWorld environment의 크기를 조절하고, environment에 존재하는 object의 수 및 positive/negative reward를 결정하는 인자의 조정을 통해 다양한 reward가 도출될 수 있도록 해야한다. Network Structure 부분에서는 kernel의 크기를 조정함으로써 효과적인 feature 추출이 가능하도록 할 수 있다.

추가적으로 수행한 VAEGAN loss 분석을 통해 발견한 것은, discriminator network의 수렴이 encoder 및 generator network의 수렴보다 훨씬 빠른 시기에 일어난다는 것이다. 이 때문에 encoder/generator의 학습이 효과적으로 이루어지지 못한다. GAN의 학습은 불안정함이 알려져있고, [13] 이를 개선하기 위해 많은 학습 전략들이 제안되어 왔다. [14, 15, 16, 17] 본 연구에서 사용된 VAEGAN 구조에 더하여, 효율적인 network 학습을 위해 제안된 방법론들을 적용해 볼 수 있다.

또, network 학습에 사용되는 loss의 추가적인 설정이 필요할 수 있다. 본 연구에서 사용된 데이터는 GAN 학습에 주로 사용되는 image input이 아니라, MDP로 정의된 강화학습 문제의 state 및 action 정보를

image 형태로 나타낸 것이다. [9]에서는 ObjectWorld MDP의 state 정보를 Fully Convolutional Network를 활용하여 해당 environment에 대응되는 reward를 output으로 내는 network를 사용하였다. 해당 학습에 사용된 loss는 단순 network loss뿐 아니라, 현재 학습된 reward와 expert의 trajectory로 추정할 수 있는 reward간의 distance를 별도의 loss로 사용하고 있다. 본 연구에서 활용한 VAEGAN은 크게 encoder loss, generator loss, discriminator loss로 나눌 수 있으나, MDP로 정의된 강화학습 문제에서 loss로 활용될 수 있는 수치를 network 학습에 적용해 볼 수 있다.

개선 방향을 거쳐 모델의 training이 성공적으로 진행되었을 때, testing data를 활용하여 관측되지 않은 state data에 대한 optimal action을 생성할 수 있는지에 대한 검증이 필요하다. 뒤이어 naive VAEGAN의 학습이 성공적으로 진행되었을 때, encoder를 통해 생성된 latent vector와 style vector를 concatenate하여 주어진 style 별 optimal action을 구분하여 생성할 수 있는지에 대한 검증을 진행해야 한다.

참 고 문 헌

- [1] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).
- [2] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.
- [3] Larsen, Anders Boesen Lindbo, et al. "Autoencoding beyond pixels using a learned similarity metric." 33rd International Conference on Machine Learning (ICML 2016) International Conference on Machine Learning. 2016.
- [4] Kingma, Diederik P., et al. "Semi-supervised learning with deep generative models." Advances in Neural Information Processing Systems. 2014.
- [5] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).
- [6] Puterman, M. L. "Markov decision processes. 1994." Jhon Wiley & Sons, New Jersey (1994).
- [7] Ng, Andrew Y., and Stuart J. Russell. "Algorithms for inverse reinforcement learning." Icml. 2000.
- [8] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.
- [9] Wulfmeier, Markus, Peter Ondruska, and Ingmar Posner. "Deep inverse reinforcement learning." CoRR, abs/1507.04888 (2015).
- [10] Levine, Sergey, Zoran Popovic, and Vladlen Koltun. "Nonlinear inverse reinforcement learning with gaussian processes." Advances in Neural Information Processing Systems. 2011.
- [11] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.

- [12] Russell, Stuart. "Learning agents for uncertain environments." Proceedings of the eleventh annual conference on Computational learning theory. ACM, 1998.
- [13] Salimans, Tim, et al. "Improved techniques for training gans." Advances in Neural Information Processing Systems. 2016.
- [14] Roth, Kevin, et al. "Stabilizing Training of Generative Adversarial Networks through Regularization." arXiv preprint arXiv:1705.09367 (2017).
- [15] Neyshabur, Behnam, Srinadh Bhojanapalli, and Ayan Chakrabarti. "Stabilizing GAN Training with Multiple Random Projections." arXiv preprint arXiv:1705.07831 (2017).
- [16] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. arXiv preprint arXiv:1506.05751, 2015.
- [17] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

약력

이 름: 김도형

생년월일: 1991년 8월 31일

전자주소: dhkim.aai@kaist.ac.kr

학력

2007. 2. – 2009. 2. 대구과학고등학교

2009. 2. – 2016. 2. 한국과학기술원 산업 및 시스템 공학과 (학사)

2016. 3. – 2018. 2. 한국과학기술원 산업 및 시스템 공학과 (석사)

경력

2016. 3. – 2016. 6. 한국과학기술원 산업 및 시스템 공학과 ”사회-경제 시스템 모델링” 조교

2016. 9. – 2016. 12. 한국과학기술원 산업 및 시스템 공학과 ”자료구조와 알고리즘 응용” 조교

2017. 3. – 2017. 10. KAIST MOOC (KOOC) 온라인 강의 ”자료구조 및 알고리즘 개론2” 조교

연구업적

1. Doyun Kim, Do-Hyeong Kim, and Il-Chul Moon. Inverse modeling of combat behavior with virtual-constructive simulation training. In Asian Simulation Conference, pages 597–606. Springer, 2016.
2. Kyungwoo Song, Do-Hyeong Kim, Su-Jin Shin, and Il-Chul Moon. Identifying the evolution of disasters and responses with network-text analysis. In Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on, pages 664–671. IEEE, 2014.
3. 박준건, 지민기, 김도형, 정요한, 이한선, 박진규, and 문일철. 강화학습을 통한 토목 공정 최적화. 한국경영과학회 학술대회논문집, pages 2172-2179, 2017.

4. 김도형 and 문일철, 레짐 변화 탐지를 활용한 동적 시뮬레이션 파라미터 교정. 한국정보과학회 학술발표논문집, pages 687-689, 2016.
5. 김기웅, 김도형, and 문일철. 시뮬레이션 분석을 위한 베이지안 푸아송 텐서 분해 방법론 활용. 한국정보과학회 학술발표논문집, pages 684-686, 2016.

