Relational Deep Reinfocement Learning

Donghyeon Kim

Seoul National University

2018-20553

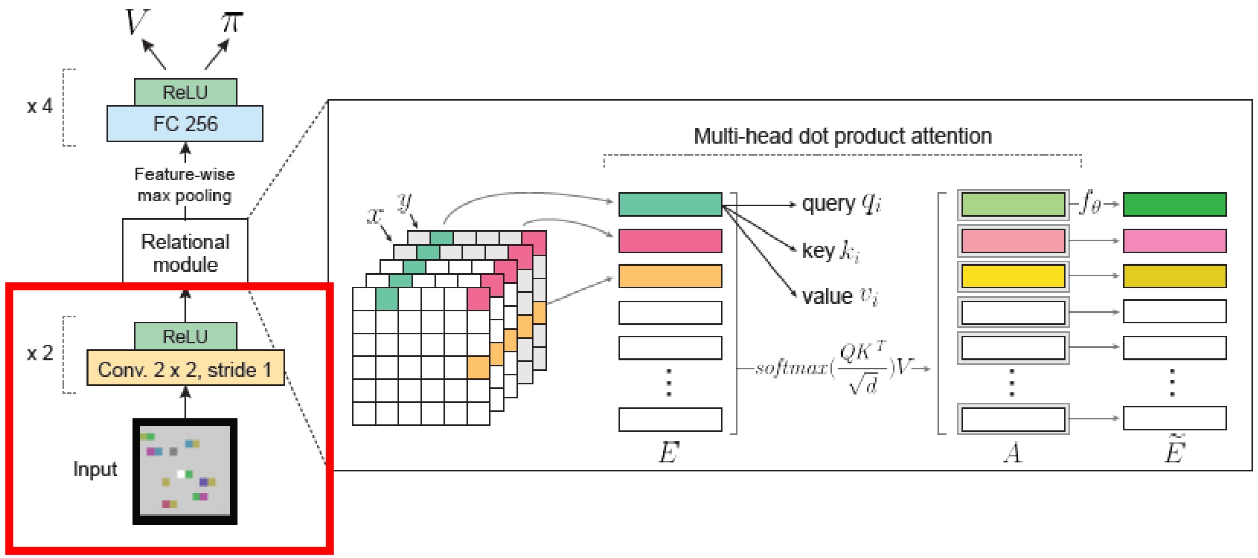
Introduction

- 기존 RL은 agent의 state(e.g vision)에서 관계가 아닌 feature에 기반하여 행동함
- 하지만 사람의 경우 vision안에 있는 물체들과 그 특성과 그 사이의 관계를 기반으로 행동함
- Current RL
    - Does not generalize to seemingly minor changes in task
    - **Overfit to the trained task**
    - Fail to learn an abstract, interpretable, and generalizable understanding of the problem
- Relational RL
    - Relational learning/Inductive logic programming + RL
    - Represent state, action, policy using a first order(relational) language ⇔ Propositional language
    - Relational language(Predicate logic) : Highly expressive, **generalization**, use of background knowledge ex) above(A,B)
    - Form architecturally specified inductive bias
    - **Perform relational reasoning via message-passing-like processing**

Introduction

- 따라서 본 논문에서는
- Use self-attention to reason about the relations between entities(pixel/object) in a scene
- Improve efficiency, generalization capacity, interpretability through structured perception and relational reasoning

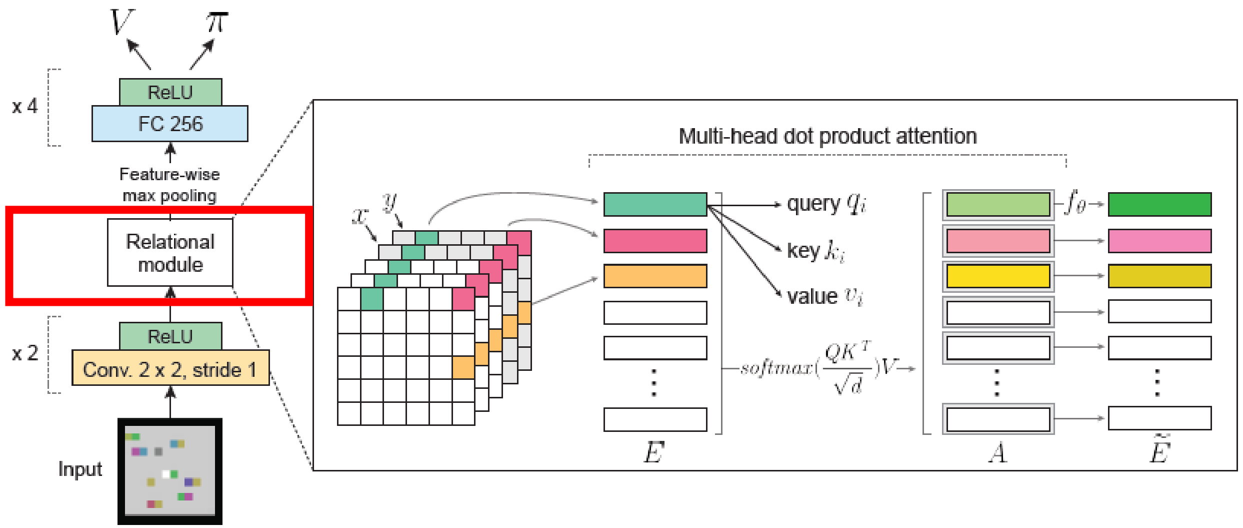
Architecture



1. Extract entities

- Minimal Assumption: Entities are things located in a particular point in space
- CNN을 통해 $n \times n$ size의 k 개 feature map 생성
- $n^2 \times k$ size entity matrix E

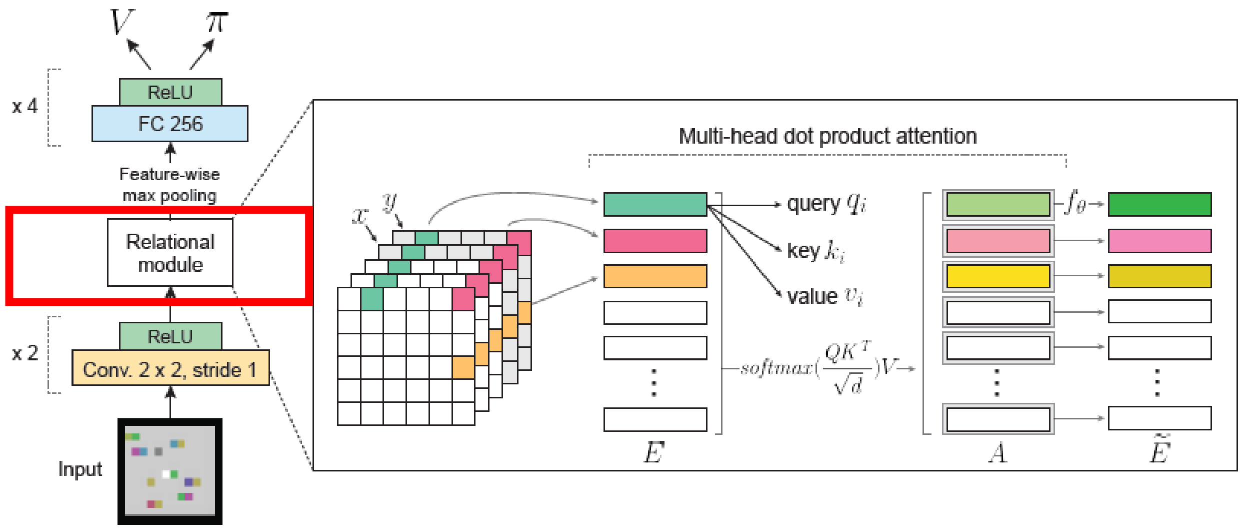
Architecture



2. Relational module(Attention block)

- Non-local computation(attention) using a shared function
- Entity-entity relations are explicitly computed when considering the messages passed between connected nodes of graph
- 여러 block을 쌓으면 higher order relation을 학습 ≈ message-passing on graph

Architecture



2. Relational module(Attention block)

- $e_{1:N}$: N $entities, q_i$: $query$ of e_i, k_i: key of e_i, v_i: $value$ of e_i

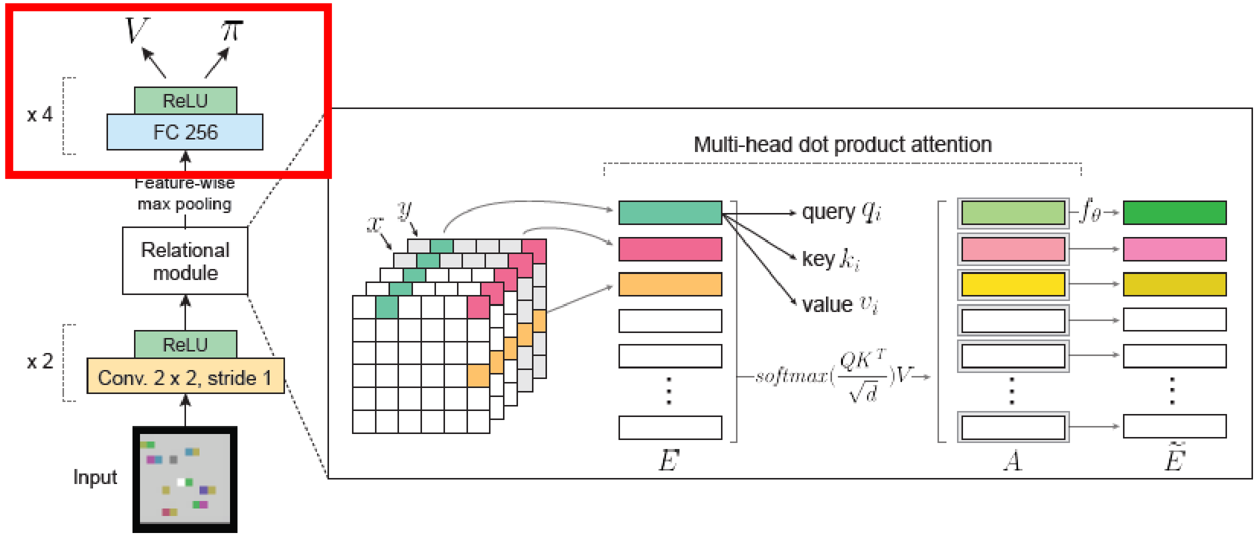
$w_{i,j} = softmax\left(\frac{q_i k_j^T}{\sqrt{d}}\right)$: $attention$ $weight$ $between$ $entity$ i and j

$a_i = \sum_{j=1:N} w_{i,j} v_j$: $cumulative$ $interaction$

$$A = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

- Use multiple attention head a_i^h : **each head have different relational semantics**
- a_i^h concatenated → passed to 2 layer MLP → residual connection with e_i → layer normalization → $\tilde{E}$

Architecture

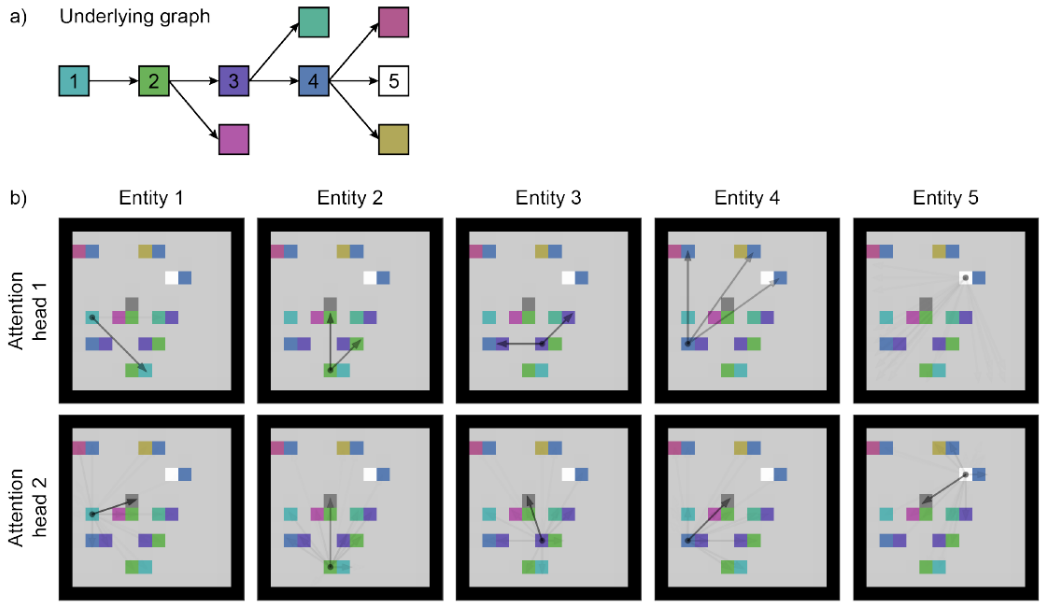


3. Actor-Critic

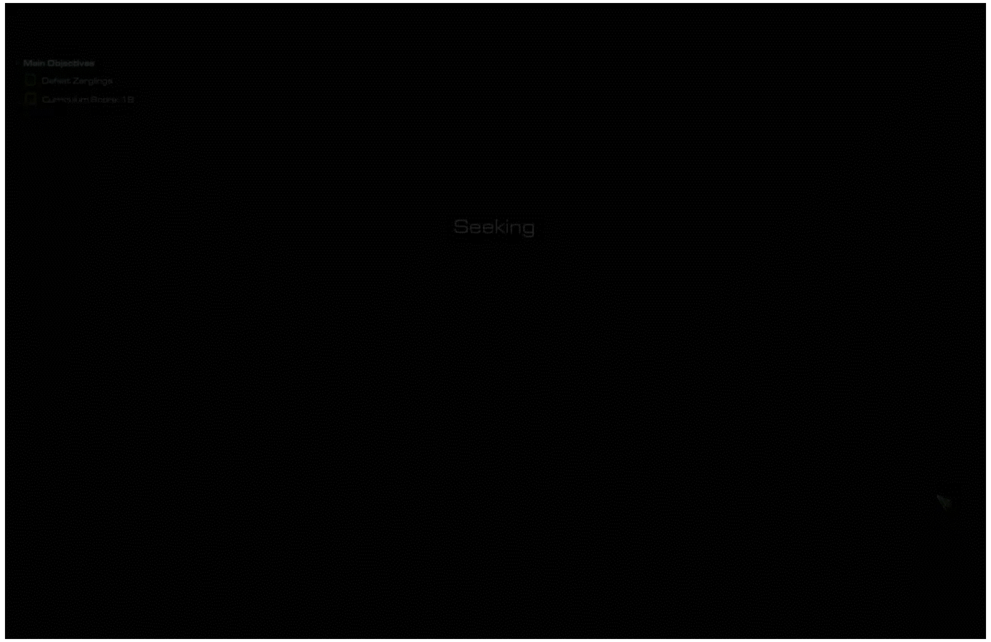
- Actor: critic의 평가 V 를 바탕으로 policy π를 update, policy에 따라 action을 선택
- Critic: actor의 policy를 평가(V),

Experiment

- Box-world와 StarCraft 2 mini-game에서 실험
- Box-world

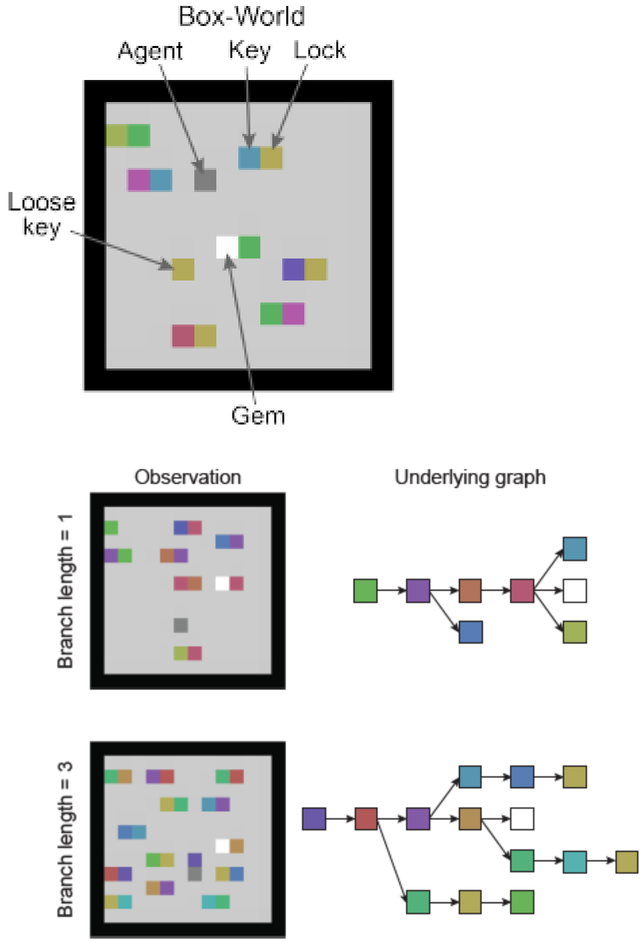


- StarCraft2



Experiment

- Box-world



✓ 회색의 Agent는 12x12 world에서 상하좌우로 움직여 key를 획득할 수 있음

✓ 최종 목표는 흰색의 Gem을 획득하는 것임

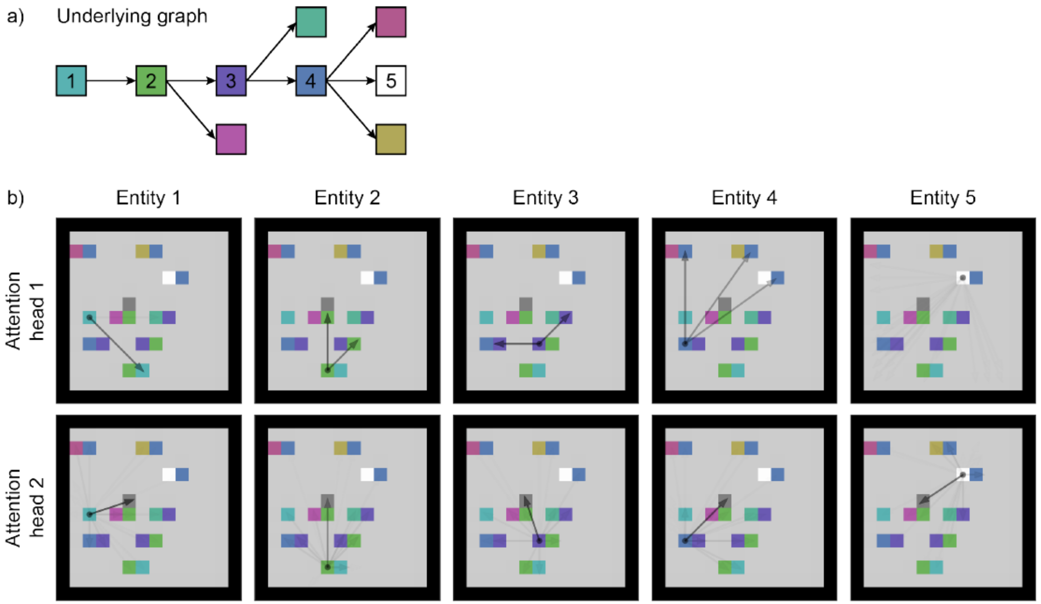
✓ 2개씩 붙어 있는 사각형들은 잠겨 있는 box들임. 오른쪽 사각형은 box를 열기 위한 key이고 왼쪽은 box를 열면 얻는 key

✓ 따라서 gem을 얻기 위한 key들은 순서를 갖고 있고 graph를 형성함

✓ Key는 한번만 사용 가능함. 따라서 Gem을 얻기 위한 올바른 순서대로 열지 않으면 dead-end로 빠지는 distractor box들이 있음

Experiment

- Box-world



✓ 제안된 attention block 사용 네트워크와 성공률 98%, 사용하지 않는 네트워크 75%

✓ Distractor branch의 길이가 길수록 더 많은 attention block들이 필요함 → 더 많은 attention block은 higher-order relation computation 가능하게 해줌

✓ 각 attention head의 attention weight은 entity간의 relation을 반영함

1. Attention head 1은 해당 source of attention로 열 수 있는 box를 point함
2. Attention head 2는 agent를 point함 → agent의 navigation역학

Experiment

- Box-world

  - **Generalization Capacity**

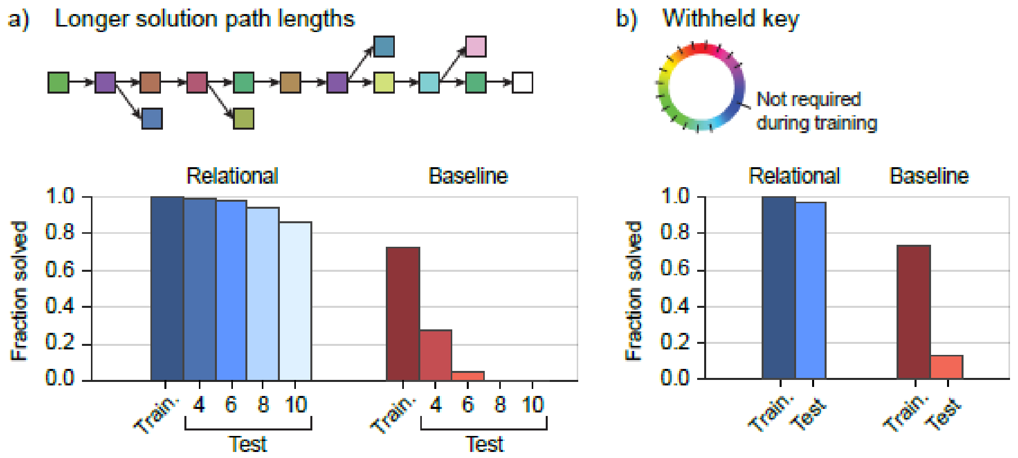
    ✓ 만약 agent가 entity간의 관계를 학습했다면(e.g. unlock(key, lock)), gem을 얻기 위한 lock의 수나 key-lock의 조합에 영향을 받지 않아야함

    ✓ 따라서 zero-shot transfer하여 (1) gem을 얻기 위한 더 긴 sequence, (2) training과정에서 본 적 없는 key-lock의 조합에 적용

    ✓ (1): Relation module 88%, without relation module 0~5%

    ✓ (2): Relation module 97%, without 13%

    ✓ 관계를 학습함으로써 기존 RL이 하지 못했던 more complex하고 previously unseen problem을 풀 수 있다.



Experiment

- StarCraft 2 mini-game

Agent	Mini-game ①	②	③	④	⑤	⑥	⑦
DeepMind Human Player [15]	26	133	46	41	729	6880	138
StarCraft Grandmaster [15]	28	177	61	215	727	7566	133
Random Policy [15]	1	17	4	1	23	12	< 1
FullyConv LSTM [15]	26	104	44	98	96	3351	6
PBT-A3C [33]	–	101	50	132	125	3345	0
Relational agent	**27**	**196** ↑	**62** ↑	**303** ↑	**736** ↑	4906	**123**
Control agent	**27**	187 ↑	61	295 ↑	602	**5055**	120

(1) Move To Beacon, (2) Collect MineralShards, (3) Find And Defeat Zerglings, (4) Defeat Roaches, (5) Defeat Zerglings And Banelings, (6)Collect Minerals And Gas, (7) Build Marines

✓ Large number of units that need to interact and collaborate

✓ Near 100 possible actions

Experiment

- StarCraft 2 mini-game

- Generalization Capacity
  - ✓ Collect mineral에서 마린 수를 2마리에서 5마리로 늘림
  - ✓ 만약 agent가 마린이 mineral을 모을 수 있는 unit이라는 관계를 학습했다면, 마린 수를 늘리는 것이 성능을 저하시키면 안됨

Code Reproduction

https://github.com/kdh0429/Relational_DRL

*본 논문을 reproducing한 https://github.com/gyh75520/Relational_DRL 를 기반으로 분석하고 약간의 코드를 추가함

Code 분석

1. BoxWorld env 코드 분석

Relational_DRL/env/gym-box-

```
# background: 0 ,1 and agent: 2
BGAndAG_COLORS = {0: [0., 0., 0.], 1: [169., 169., 169.],
                  2: [105., 105., 105.]}
# gem: 3
CorrectBox_COLORS = {3: [255., 255., 255.],
                     4: [0., 255., 0.], 5: [255.0, 0., 0.],
                     6: [255., 0., 255.], 7: [255., 255., 0.]}

DistractorBox_COLORS = {8: [0., 255., 255.], 9: [255.0, 127.5, 127.5],
                        10: [127.5, 0., 255.], 11: [255., 127.5, 0.],
                        12: [127.5, 127.5, 255.], 13: [0., 127.5, 127.5],
                        14: [127.5, 127.5, 0.], 15: [255., 0., 127.5], }

COLORS = dict(list(BGAndAG_COLORS.items()) + list(CorrectBox_COLORS.items()) + list(DistractorBox_COLORS.items()))

# branch_length = 1
EASY_BOX_LIST = [(7, 6), (4, 7), (5, 4), (3, 5), (8, 7), (9, 5), (10, 5)]
EASY_END_LIST = [8, 9, 10]
# branch_length = 2
MEDIUM_BOX_LIST = [(7, 6), (4, 7), (5, 4), (3, 5), (8, 7), (11, 8), (9, 4), (12, 9), (10, 5), (13, 10)]
MEDIUM_END_LIST = [11, 12, 13]
# branch_length = 3
HARD_BOX_LIST = [(7, 6), (4, 7), (5, 4), (3, 5), (8, 7), (11, 8), (9, 4), (12, 9), (14, 12), (10, 5), (13, 10), (15, 13)]
HARD_END_LIST = [11, 14, 15]
BOX_DICT = {'easy': EASY_BOX_LIST, 'medium': MEDIUM_BOX_LIST, 'hard': HARD_BOX_LIST}
END_DICT = {'easy': EASY_END_LIST, 'medium': MEDIUM_END_LIST, 'hard': HARD_END_LIST}
```

환경/agent 색 정의

Gem을 얻기 위한 box 색 정의

Gem을 얻기 헷갈리게 하는(distractor) box 색 정의

난이도에 따라 branch 길이
및 distractor 수 변화

Code 분석

1. BoxWorld env 코드 분석

Relational_DRL/env/gym-box-world/gym_boxworld/envs/**Box_world_rand_env.py**

class BoxWoldRandEnv

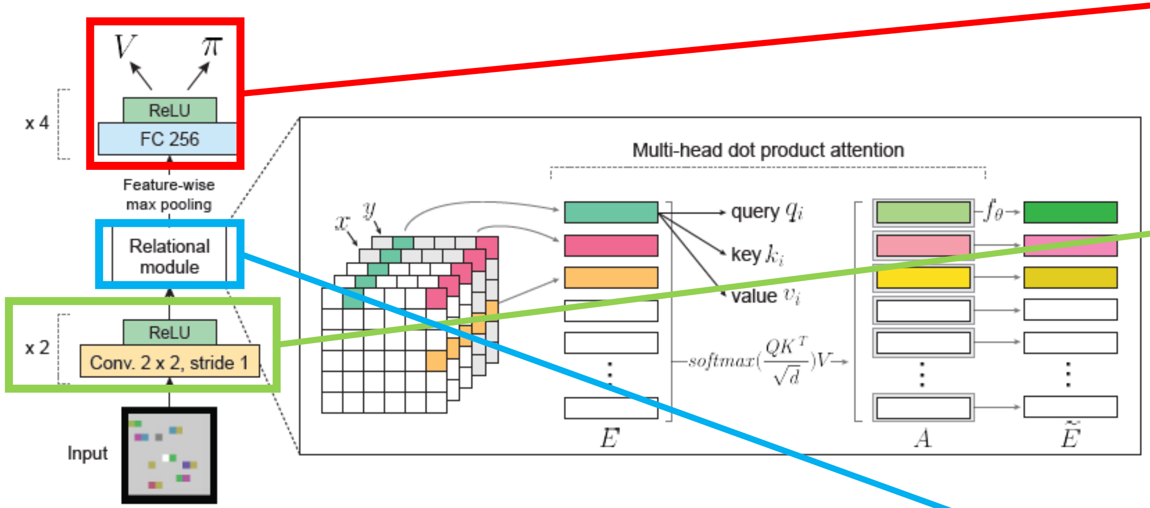
ㄴ __init__(self, level):	action 정의 및 14x14 환경 정의
ㄴ set_box(self, box):	box 생성
ㄴ reset(self):	환경 reset
ㄴ seed(self, seed=None):	random seed 설정
ㄴ step(self, action):	action에 따라 agent 이동 및 box 열기
ㄴ _update_key(self, nxt_color):	현재 보유 key 업데이트
ㄴ _agent_move(self, next_agent_state):	agent 이동
ㄴ _get_agent(self, world_map):	agent state 반환
ㄴ get_current_agent_position(self):	agent 현재 위치 반환
ㄴ _read_world_map(self, path):	env map 생성
ㄴ _worldmap_to_obervation(self, world_map):	map을 이미지로 변환
ㄴ render(self, model='human'):	환경 render
ㄴ get_action_meanings(self):	action의 의미(상하좌우) 반환

Code 분석

2. Relational Policy 코드 분석

Relational_DRL/relational_policies.py

```python
class RelationalPolicy(ActorCriticPolicy):
    def __init__(self, sess, ob_space, ac_space, n_env, n_steps, n_batch, reuse=False, net_arch=None,
                 act_fun=tf.tanh, feature_extraction="cnn", **kwargs):
        super(RelationalPolicy, self).__init__(sess, ob_space, ac_space, n_env, n_steps, n_batch, reuse=reuse,
                                               scale=(feature_extraction == "cnn"))
        self._kwargs_check(feature_extraction, kwargs)
        with tf.variable_scope("model", reuse=reuse):
            print('self.processed_obs', self.processed_obs)
            relation_block_output = self.relation_block(self.processed_obs)
            pi_latent = vf_latent = tf.layers.flatten(relation_block_output)
            # original code
            self._value_fn = linear(vf_latent, 'vf', 1)
            self._proba_distribution, self._policy, self.q_value = \
                self.pdtype.proba_distribution_from_latent(pi_latent, vf_latent, init_scale=0.01)
```

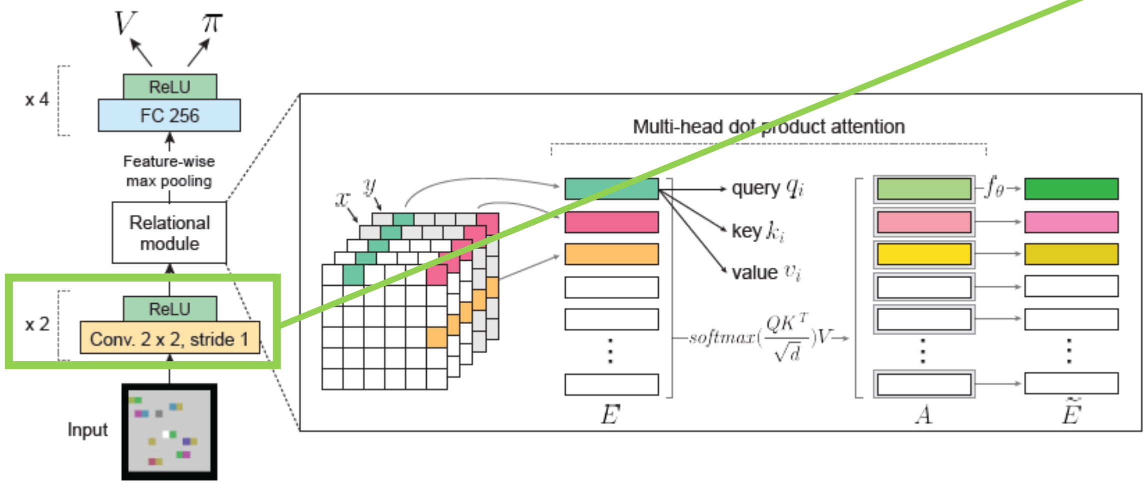



```python
    def relation_block(self, processed_obs):
        entities = build_entities(processed_obs, self.reduce_obs)
        print('entities:', entities)
        # [B,n_heads,N,Deepth=D+2]
        MHDPA_output, self.relations = MHDPA(entities, n_heads=2)
        print('MHDPA_output', MHDPA_output)
        # [B,n_heads,N,Deepth]
        residual_output = residual_block(entities, MHDPA_output)
        print('residual_output', residual_output)
        # max_pooling [B,n_heads,N,Deepth] --> [B,n_heads,Deepth]
        maxpooling_output = tf.reduce_max(residual_output, axis=2)
        print('maxpooling_output', maxpooling_output)
        # [B,n_heads*Deepth]
        # output = tf.layers.flatten(maxpooling_output)
        # output = layerNorm(output, "relation_layerNorm")
        # print('relation_layerNorm', output)
        return maxpooling_output
```

Code 분석

2. Relational Policy 코드 분석

Relational_DRL/utils.py



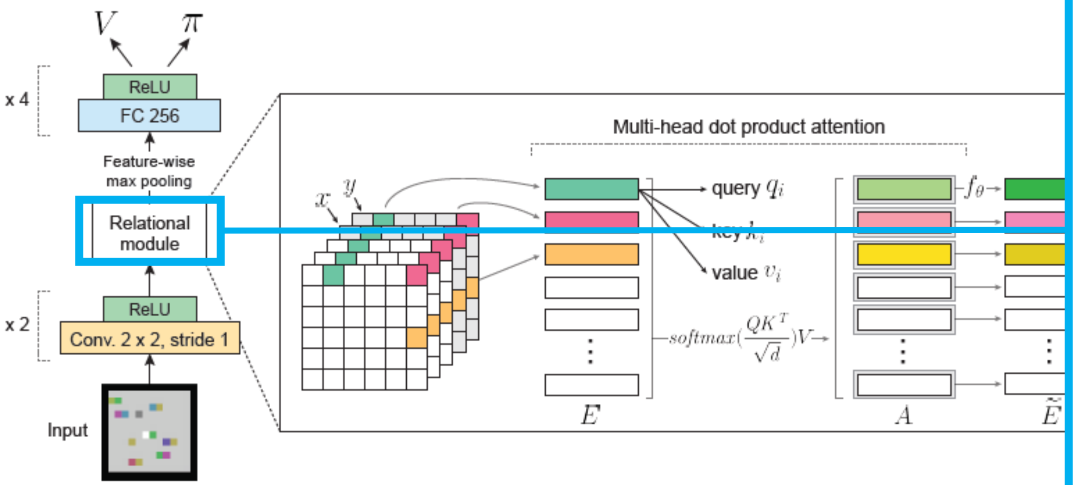
Convolution layer를 거쳐 Entity 생성

```
def build_entities(processed_obs, reduce_obs=False):
    coor = get_coor(processed_obs)
    cnn_extractor = deepconcise_cnn
    if reduce_obs:
        # [B,Height,W,D+2]
        processed_obs = tf.concat([processed_obs, coor], axis=3)
        # [B,N,D] N=Height*w+1
        entities = reduce_border_extractor(processed_obs, cnn_extractor)
    else:
        # # [B,Height,W,D]
        # extracted_features = cnn_extractor(processed_obs)
        # # [B,Height,W,D+2]
        # entities = tf.concat([extracted_features, coor], axis=3)
        # # [B,N,D] N=Height*w
        # entities = entities_flatten(entities)

        # [B,Height,W,D+2]
        processed_obs = tf.concat([processed_obs, coor], axis=3)
        # [B,Height,W,D]
        entities = cnn_extractor(processed_obs)
        # [B,N,D] N=Height*w
        entities = entities_flatten(entities)

    return entities
```

Code 분석

2. Relational Policy 코드 분석

Relational_DRL/utils.py



Relational Module
: Multi Head Dot Product Attention

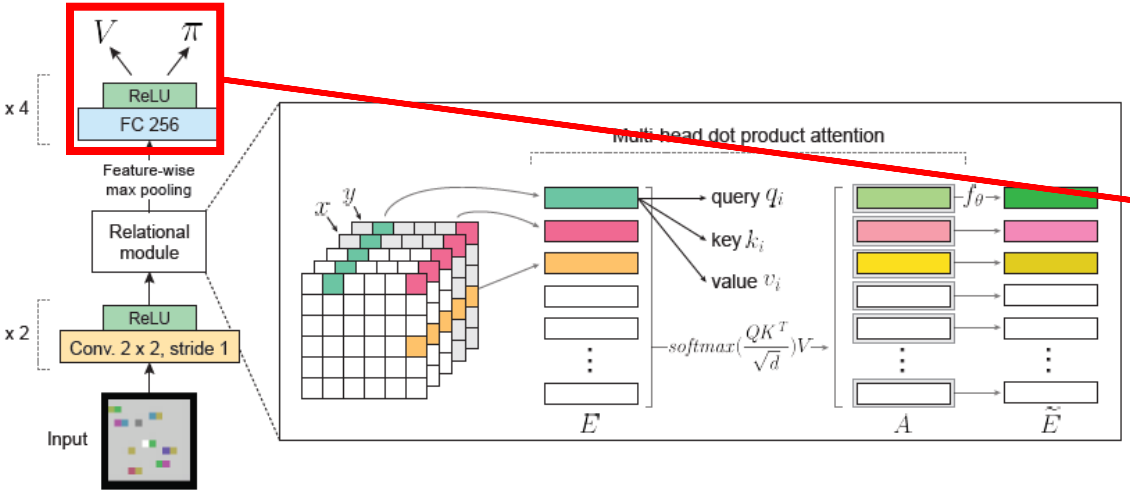
```python
def MHDPA(entities,  n_heads):
    """
    An implementation of the Multi-Head Dot-Product Attention architecture in "Relational Deep Reinforcement Learning"
    https://arxiv.org/abs/1806.01830
    ref to the RMC architecture on https://github.com/deepmind/sonnet/blob/master/sonnet/python/modules/relational_memory.py

    :param entities: (TensorFlow Tensor) entities [B,N,D]
    :param n_heads: (float) The number of attention heads to use
    :return: (TensorFlow Tensor) [B,n_heads,N,D]
    """
    q, k, v = getQKV(entities, n_heads, 'QKV')
    # dot_product *= qkv_size ** -0.5
    # [B,n_heads,N,N]
    dot_product = tf.matmul(q, k, transpose_b=True)
    channels = v.shape[-1].value
    dot_product = dot_product * (channels**-0.5)
    return updateRelations(dot_product, v)
```

각 Entity의 query, key, value를 통해 attention weight인 $softmax\left(\frac{QK^T}{\sqrt{d}}\right)$를 계산하고 cumulative interaction $A = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$ 계산

Code 분석

2. Relational Policy 코드 분석

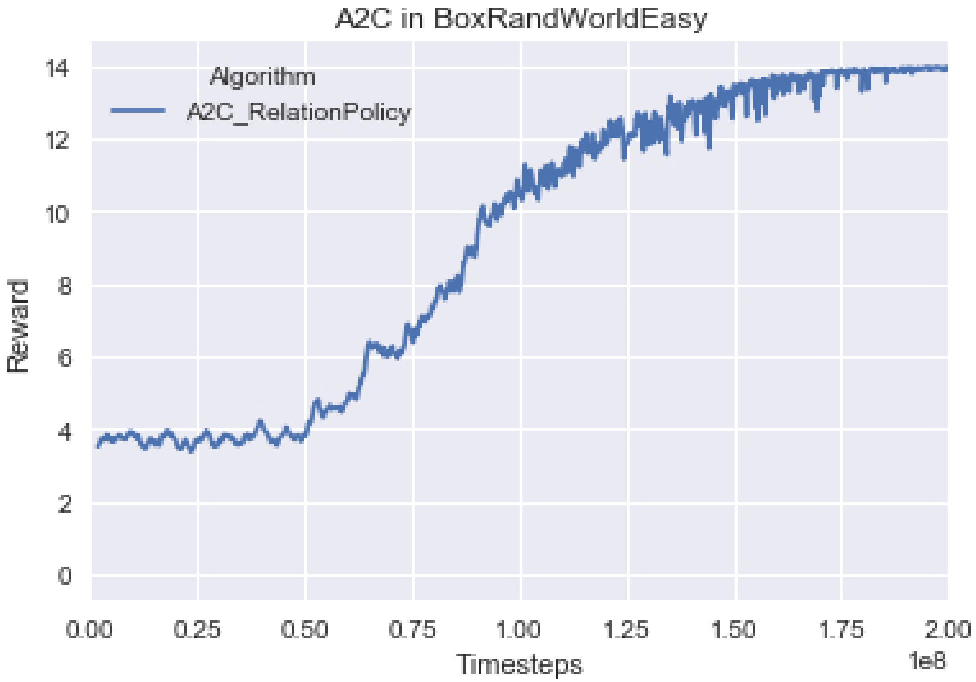
Relational_DRL/relational_policies.py

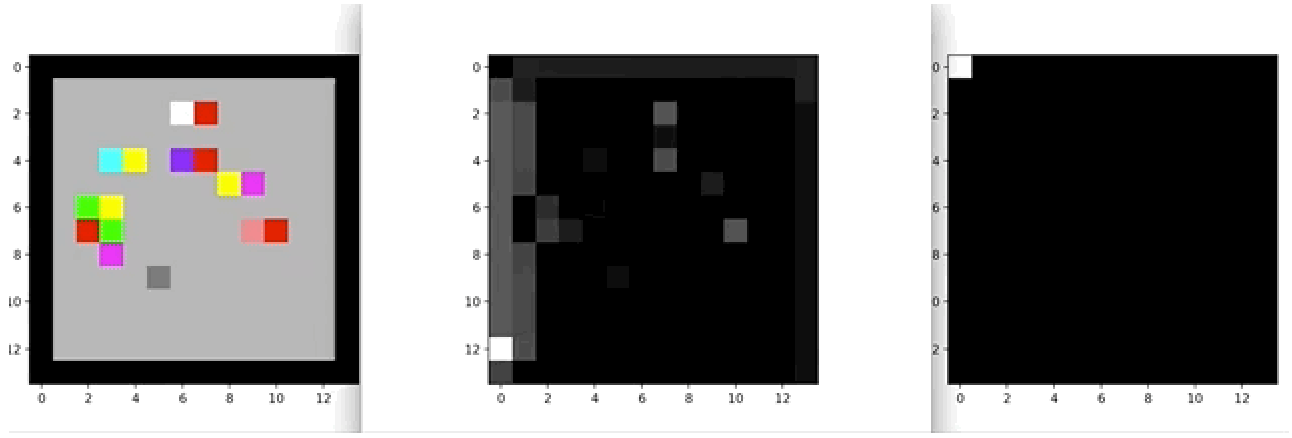


Relation Module의 output(latent)로부터 policy와 value function network 생성

```python
class RelationalPolicy(ActorCriticPolicy):
    def __init__(self, sess, ob_space, ac_space, n_env, n_steps, n_batch, reuse=False, net_arch=None,
                 act_fun=tf.tanh, feature_extraction="cnn", **kwargs):
        super(RelationalPolicy, self).__init__(sess, ob_space, ac_space, n_env, n_steps, n_batch, reuse=reuse,
                                               scale=(feature_extraction == "cnn"))
        self._kwargs_check(feature_extraction, kwargs)
        with tf.variable_scope("model", reuse=reuse):
            print('self.processed_obs', self.processed_obs)
            relation_block_output = self.relation_block(self.processed_obs)
            pi_latent = vf_latent = tf.layers.flatten(relation_block_output)
            # original code
            self._value_fn = linear(vf_latent, 'vf', 1)
            self._proba_distribution, self._policy, self.q_value = \
                self.pdtype.proba_distribution_from_latent(pi_latent, vf_latent, init_scale=0.01)
```

실험 결과





Head 1

Head 2

Relational Policy로 학습한 결과

Multi Head(2개)에서 agent가 attention하고 있는 entity 시각화