MAPPO

The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games

Index

- Introduction
- Agent Setting
- Experiment setup
- Experiment Result
- Factors influential to PPO’s performance
  - Value Normalization
  - Input Representation to Value Function
  - Empirical Analysis
  - Training Data Usage
  - PPO Clipping
  - PPO Batch Size
- Conclusion

Introduction

- The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games,
  2021 BAIR 쪽 arXiv 등록 논문, 인용수 129
- MARL 환경에서 큰 성공(Alpha Star, OpenAI Five) 등이 대규모 리소스를 바탕으로
  on-policy 알고리즘 활용
- Academic Level 에서는 대규모 리소스를 활용할 수 없기 때문에 sample efficient 를 위해
  off-policy 알고리즘 위주 연구 진행
- 본 논문에서 MARL 에 PPO 가 효과적인 성능을 발휘할 수 있는 factor 들과 실험결과 제시

※ 주의 : MARL 을 위한 PPO 알고리즘의 변형 혹은 개선이 아닌 MARL 환경에서 잘 동작하기 위한 Tuning 요소와 실험결과 제시

Agent Setting

- Decentralized-POMDP 가정

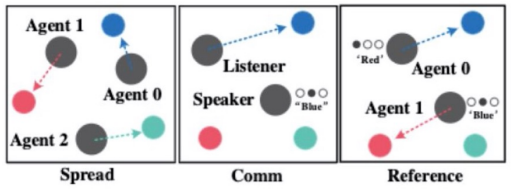
$$\langle \mathcal{S}, \mathcal{A}, O, R, P, n, \gamma \rangle.$$

  - O : global state s에 대한 I 번째 agent 의 local observation O (s, i)
  - P : trasition (s' | s, a)
  - R : shared reward function R (s, a)

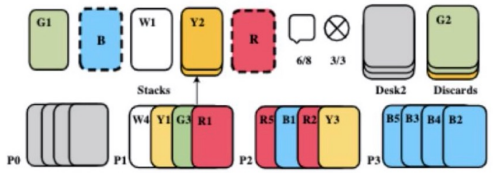
- PPO 구조를 거의 그대로 활용하고 Value Function 만 Centralized value function 활용하는 형태 (CTDE)

Experiment setup

- On-policy : MAPPO (Multi-Agent PPO), IPPO(Independent PPO)
  off-policy : MADDPG, QMix, RODE, QPlex, SAD 간 비교
- RL env : MPE, SMAC, Hanabi, GRF 활용



(a) MPE scenarios



(b) 4-player Hanabi-Full

(c) SMAC corridor

(d) SMAC 2c_vs_64zg

- Hardware : 64-core CPU, 128 RAM for experiments, RTX 3090 GPU 1EA for training

Experiment Results

- MPE Result

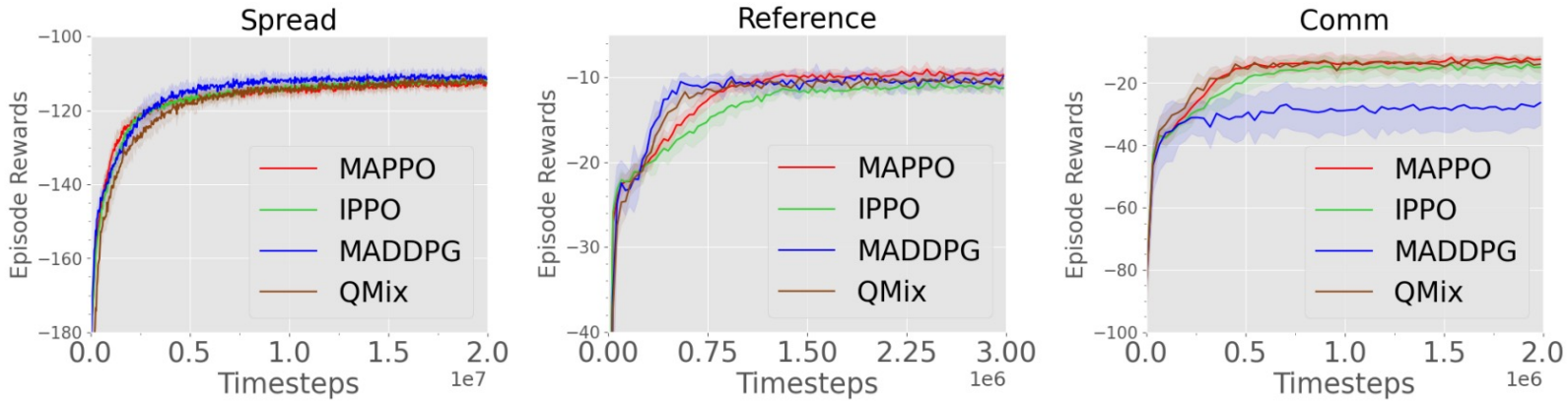


Figure 1: Performance of different algorithms in the MPEs.

- MAPPO, IPPO, MADDPG, QMix 결과 비교
- MPE 에서는 global state 를 제공하지 않기 때문에 local observation 을 concat 하여 활용
- 10개 이상 seed 실험 결과 비교
- MAPPO 결과가 MADDPG, QMix 와 비슷하거나 상회하고 IPPO 보다 조금 더 일찍 수렴

Experiment Results

- SMAC Result

Map	MAPPO(FP)	MAPPO(AS)	IPPO	QMix	RODE*	MAPPO*(FP)	MAPPO*(AS)
2m vs_1z	**100.0**(0.0)	**100.0**(**0.0**)	**100.0**(0.0)	**95.3**(5.2)	/	*100.0*(0.0)	*100.0*(0.0)
3m	**100.0**(0.0)	**100.0**(1.5)	**100.0**(0.0)	96.9(1.3)	/	*100.0*(0.0)	*100.0*(1.5)
2svs1sc	**100.0**(0.0)	**100.0**(0.0)	**100.0**(1.5)	96.9(2.9)	*100.0*(0.0)	*100.0*(0.0)	*100.0*(0.0)
2s3z	**100.0**(0.7)	**100.0**(1.5)	**100.0**(0.0)	95.3(2.5)	*100.0*(0.0)	96.9(1.5)	96.9(1.5)
3svs3z	**100.0**(0.0)	**100.0**(0.0)	**100.0**(0.0)	**96.9**(12.5)	/	*100.0*(0.0)	*100.0*(0.0)
3svs4z	**100.0**(1.3)	**98.4**(1.6)	**99.2**(1.5)	**97.7**(1.7)	/	*100.0*(2.1)	*100.0*(1.5)
so many baneling	**100.0**(0.0)	**100.0**(0.7)	**100.0**(1.5)	96.9(2.3)	/	*100.0*(1.5)	96.9(1.5)
8m	**100.0**(0.0)	**100.0**(0.0)	**100.0**(0.7)	97.7(1.9)	/	*100.0*(0.0)	*100.0*(0.0)
MMM	**96.9**(0.6)	93.8(1.5)	**96.9**(0.0)	**95.3**(2.5)	/	*93.8*(2.6)	*96.9*(1.5)
1c3s5z	**100.0**(0.0)	96.9(2.6)	**100.0**(0.0)	96.1(1.7)	*100.0*(0.0)	*100.0*(0.0)	96.9(2.6)
bane vs bane	**100.0**(0.0)	**100.0**(0.0)	**100.0**(0.0)	**100.0**(0.0)	*100.0*(46.4)	*100.0*(0.0)	*100.0*(0.0)
3svs5z	**100.0**(0.6)	**99.2**(1.4)	**100.0**(0.0)	**98.4**(2.4)	78.9(4.2)	*98.4*(5.5)	*100.0*(1.2)
2cvs64zg	**100.0**(0.0)	**100.0**(0.0)	98.4(1.3)	92.2(4.0)	*100.0*(0.0)	*96.9*(3.1)	95.3(3.5)
8mvs9m	**96.9**(0.6)	**96.9**(0.6)	**96.9**(0.7)	92.2(2.0)	/	*84.4*(5.1)	*87.5*(2.1)
25m	**100.0**(1.5)	**100.0**(4.0)	**100.0**(0.0)	85.9(7.1)	/	*96.9*(3.1)	*93.8*(2.9)
5mvs6m	**89.1**(2.5)	**88.3**(1.2)	**87.5**(2.3)	75.8(3.7)	*71.1*(9.2)	*65.6*(14.1)	*68.8*(8.2)
3s5z	**96.9**(0.7)	**96.9**(1.9)	**96.9**(1.5)	88.3(2.9)	*93.8*(2.0)	71.9(11.8)	53.1(15.4)
10mvs11m	**96.9**(4.8)	**96.9**(1.2)	**93.0**(7.4)	**95.3**(1.0)	*95.3*(2.2)	81.2(8.3)	*89.1*(5.5)
MMM2	**90.6**(2.8)	**87.5**(5.1)	**86.7**(7.3)	**87.5**(2.6)	*89.8*(6.7)	51.6(21.9)	28.1(29.6)
3s5zvs3s6z	**84.4**(34.0)	63.3(19.2)	**82.8**(19.1)	**82.8**(5.3)	*96.8*(25.11)	*75.0*(36.3)	18.8(37.4)
27mvs30m	**93.8**(2.4)	85.9(3.8)	69.5(11.8)	39.1(9.8)	*96.8*(1.5)	*93.8*(3.8)	*89.1*(6.5)
6hvs8z	**88.3**(3.7)	**85.9**(30.9)	**84.4**(33.3)	9.4(2.0)	*78.1*(37.0)	*78.1*(5.6)	*81.2*(31.8)
corridor	**100.0**(1.2)	**98.4**(0.8)	**98.4**(3.1)	84.4(2.5)	*65.6*(32.1)	*93.8*(3.5)	*93.8*(2.8)

- MAPPO(FP), MAPPO(AS), IPPO, RODE, QMix win rate median 값 비교
- FP 는 Feature Pruned input, AS 는 Agent-specific centralized input
- 같은 training sample 을 사용한 RODE(*표시) 와 비교 했을 때 비슷하거나 상회

Experiment Results

- Hanabi Result

# Players	Metric	MAPPO	IPPO	SAD	VDN
2	Avg.	23.89(0.02)	**24.00**(0.02)	23.87(0.03)	23.83(0.03)
	Best	**24.23**(0.01)	24.19(0.02)	24.01(0.01)	23.96(0.01)
3	Avg.	**23.77**(0.20)	23.25(0.33)	23.69(0.05)	23.71(0.06)
	Best	**24.01**(0.01)	23.87(0.03)	23.93(0.01)	23.99(0.01)
4	Avg.	**23.57**(0.13)	22.52(0.37)	23.27(0.26)	23.03(0.15)
	Best	23.71(0.01)	23.06(0.03)	**23.81**(0.01)	23.79(0.00)
5	Avg.	**23.04**(0.10)	20.75(0.56)	22.06(0.23)	21.28(0.12)
	Best	**23.16**(0.01)	22.54(0.02)	23.01(0.01)	21.80(0.01)

Table 2: Best and Average evaluation scores of MAPPO, IPPO, SAD, and VDN on Hanabi-Full. Results are reported over at-least 3 seeds.

- MAPPO, IPPO, SAD, VDN 간 최소 3 seed, 10B 정도 step 학습 결과 비교
- Hanabi 에서 좋은 성능을 보인다는 것은 다른 player 의 action 으로 부터 intent 를 잘 파악한다는 의미
- Player 가 늘어날 수록 IPPO 보다 MAPPO 가 좋은 성능을 보임

Experiment Results

- GRF Result

Scen.	MAPPO	QMix	CDS	TiKick
3v.1	**88.03**(1.06)	8.12(2.83)	76.60(3.27)	76.88(3.15)
CA(easy)	**87.76**(1.34)	15.98(2.85)	63.28(4.89)	/
CA(hard)	**77.38**(4.81)	3.22(1.60)	58.35(5.56)	73.09(2.08)
Corner	**65.53**(2.19)	16.10(3.00)	3.80(0.54)	33.00(3.01)
PS	**94.92**(0.68)	8.05(3.66)	**94.15**(2.54)	/
RPS	**76.83**(1.81)	8.08(4.71)	62.38(4.56)	79.12(2.06)

Table 3: Average evaluation success rate and standard deviation (over six seeds) on GRF scenarios for different methods. All values within 1 standard deviation of the maximum success rate are marked in bold. We separate TiKick from the other methods as it uses pretrained models and thus does not constitute a direct comparison.

- MAPPO(IPPO), QMix, CDS, Tikick 간 25M ~ 50M 학습 결과 비교
- 거의 모든 task 에서 MAPPO 가 우세

Factors influential to PPO's performance

- Value Normalization
  - 학습 환경에 따라 실제 받는 reward 편차가 클 수 있기 때문에 Value Normalization 진행 (0 to 1)
  - 실제로 MPE Spread 환경은 리워드 range 가 -200 ~ 0 으로 Value Normalization 효과가 좋았음

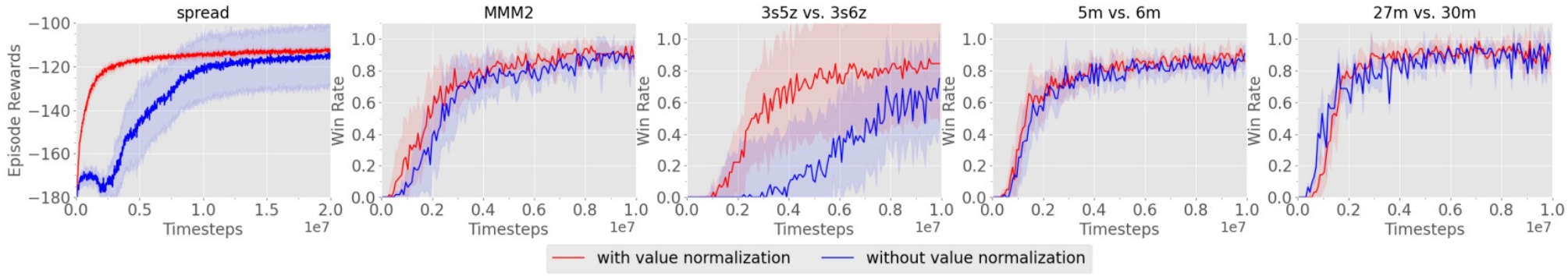


Figure 2: Impact of value normalization on MAPPO's performance in SMAC and MPE.

Factors influential to PPO's performance

- Input Representation to Value Function
  - 전형적으로 2가지 유형의 Global State 를 CTDE 구조의 Value Function input 으로 활용
    - Concatenation of local observation(CL) :
      local agent observation 을 concat, unobserved 정보 생략 가능성 존재
    - Environment Provided global state(EP) :
      environment 에서 주어지는 global state, 중요한 agent specific 정보 생략 가능성 존재
  - CL 과 EP 약점을 극복하기 위해 EP + agent specific info 인 Agent-Specific Global State(AS) 에 불필요한 정보 중복을 제거한 Featured-Pruned Agent-Specific Global State(FP) 활용

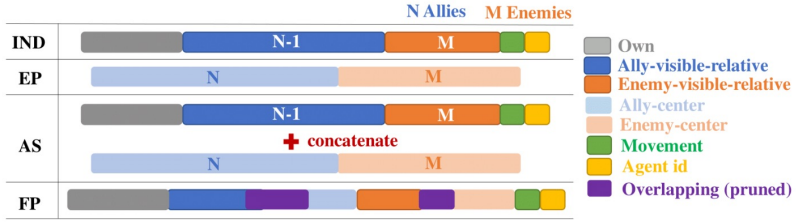


Figure 3: Different value function inputs with example features contained in each state (SMAC-specific). *IND* refers to using decentralized inputs (agents' local observations), *EP* refers to the environment provided global state, *AS* is an agent-specific global state which concatenates *EP* and *IND*, and *FP* is an agent-specific global state which prunes overlapping features from *AS*. *EP* omits important local data such as agent ID and available actions.

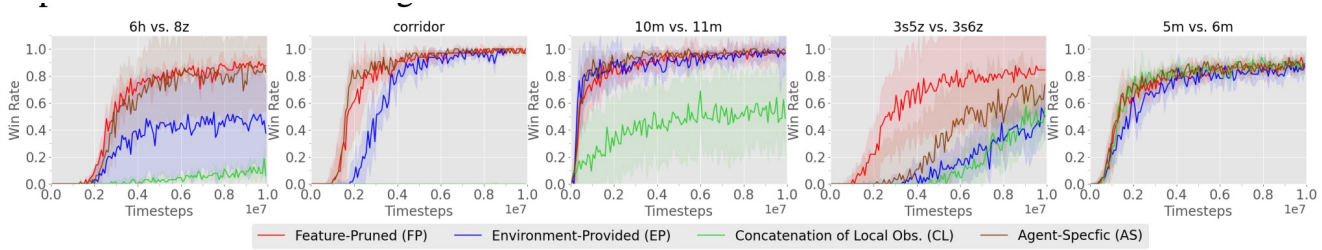
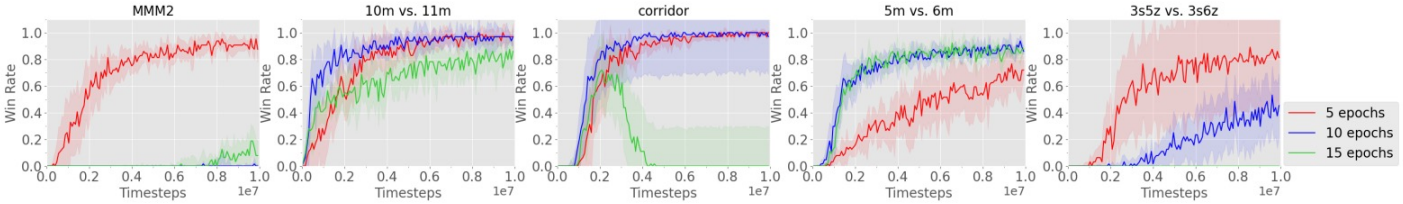


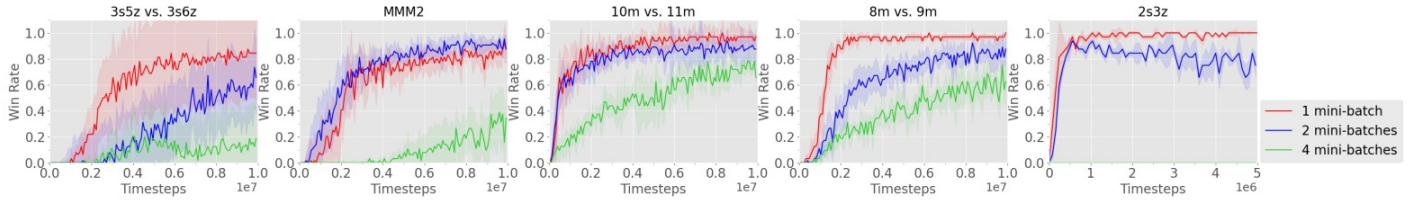
Figure 4: Effect of different value function input representations (described in Fig. 3).

Factors influential to PPO's performance

- Training Data Usage
  - 일반적으로 Single-agent continuous control domain 에서 10 epoch 동안 sample data 를 32~ 64 mini-batch 로 쪼개서 사용
  - Multi-agent domain 에서 sample data 를 자주 재사용하면 오히려 성능이 떨어지는 것을 확인
  - Easy task 의 경우 15 epoch, difficult task 의 경우 5 ~ 10 epoch 및 최대 2 mini-batch 로 쪼개는 것이 적절

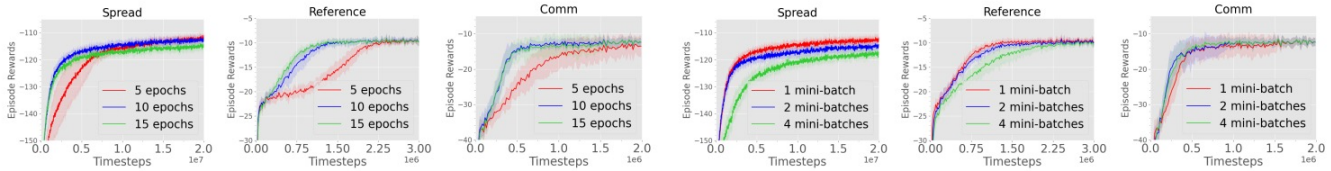


(a) effect of different training epochs.



(b) effect of different mini-batch numbers.

Figure 5: Effect of epoch and mini-batch number on MAPPO's performance in SMAC.



(a) effect of different training epochs. (b) effect of different mini-batch numbers.

Figure 6: Effect of epoch and mini-batch number on MAPPO's performance in MPE.

Factors influential to PPO's performance

- PPO Clipping
  - PPO 에서는 Training iter 마다 너무 큰 policy, value 변화를 방지하기 위해 clipping 사용
  - Multi-agent 에서도 마찬가지로 모든 agent 에 대해 policy, value 변화가 불안정하게 변하는 것을 막기 위해 작은 값의 ε 사용 (under 0.2)

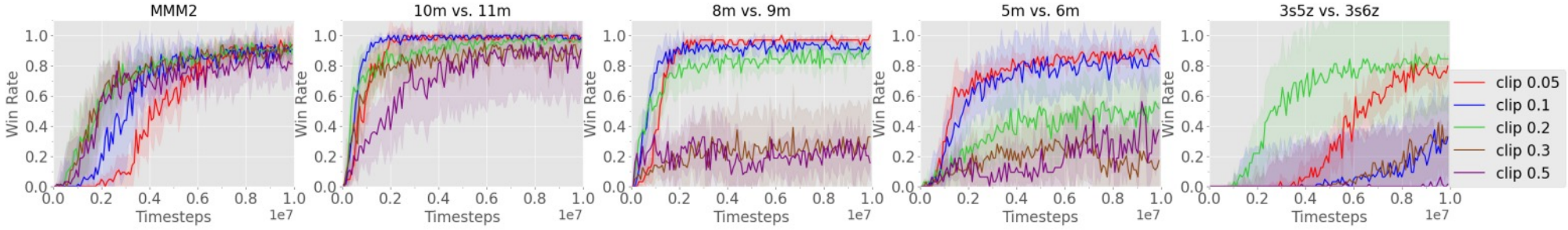
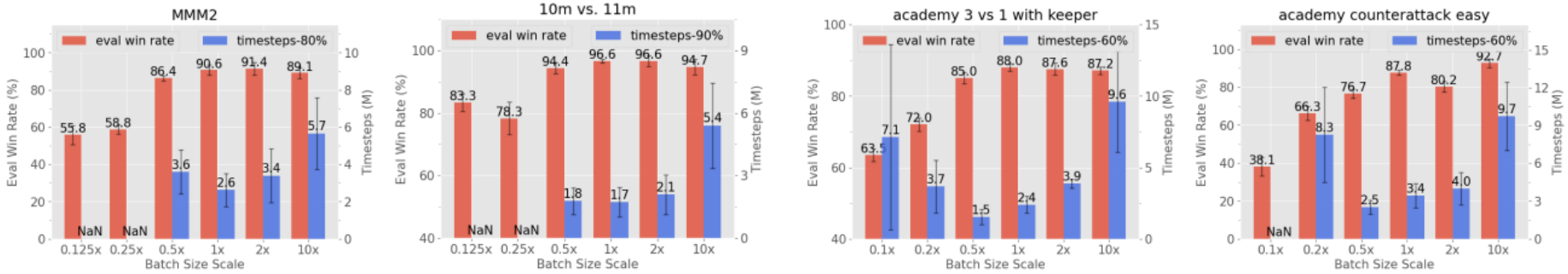


Figure 7: Effect of different clipping strengths on MAPPO's performance in SMAC.

Factors influential to PPO's performance

- PPO Batch Size
  - On-policy 의 경우 batch size 가 크면 gradient 를 더 정확하게 계산이 가능하지만 많은 리소스 필요
  - Sample-efficiency 를 잘 고려하여 큰 batch-size 활용



(a) SMAC (b) GRF

Figure 8: Effect of batch size on MAPPO's performance in SMAC and GRF. Red bars show the final win-rates. The blue bars show the number of environment steps required to achieve a strong win-rate (80% or 90% in SMAC and 60% in GRF) as a measure of sample efficiency. "NaN" means such a win-rate was never reached. The x-axis specifies the batch-size as a multiple of the batch-size used in our main results. A sufficiently large batch-size is required to achieve the best final performance/sample efficiency; further increasing the batch size may hurt sample efficiency.

Conclusion

- MAPPO 라는 단어에 무색하게 기존 PPO 대비 극적으로 새로운 점을 발견하지 못함
- Discrete action, All cooperative env, homogeneous agents 등 제한적 환경
- Tuning Factor 역시 조금은 일반적인 요소들을 언급

하지만…

- 학습 시 오히려 간과하기 쉬운 일반적인 요소를 명시적으로 제시
- 이론적으로 설명하기 어려운 체득의 결과를 무수히 많은 실험을 통해 공유