强化学习原理及应用
Reinforcement Learning (RL): Theories & Applications

DCS6289 Spring 2022

Yucong Zhang （张宇聪）

School of Computer Science and Engineering
Sun Yat-Sen University

Lecture 10：Multi-Agent RL

17th May. 2022

Multi-Agent RL

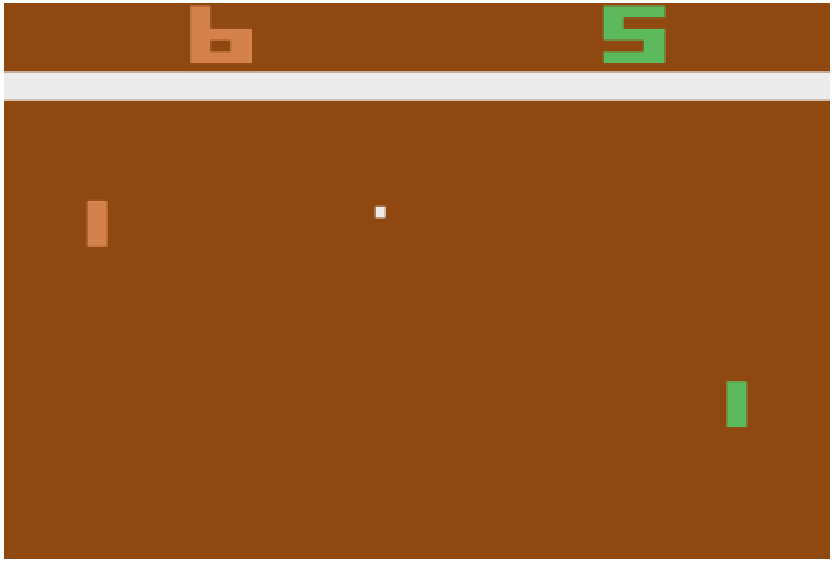
- Independent Learning
  - IQL
  - IL with parameter sharing

- Learning cooperation
  - MADDPG
  - COMA
  - VDN
  - QMIX

Multi-Agent RL

Independent Learning

- Evaluate single-agent DRL algorithm in multiagent settings.
- IQL
  - ✓ Study emergent cooperative and competitive strategies between multiple agents controlled by autonomous deep Q-Networks.
  - ✓ Use Atari video games as the environment.
  - ✓ Explore how two agents behave and interact in complex. environment when trained with different rewarding schemes.



Rewarding Schemes

1. Fully Competitive
2. Fully Cooperative
3. Transition between Cooperation and Competition

Tampuu, Ardi, et al. "Multiagent cooperation and competition with deep reinforcement learning." *PloS one* 12.4 (2017): e0172395.

Multi-Agent RL

IQL

- Score More than the Opponent (Fully Competitive)
  - The player who last touches the outgoing ball gets a plus point, and the player losing the ball a minus point.

	Left player scores	Right player scores
Left player reward	$+1$	-1
Right player reward	-1	$+1$

- Loosing the Ball Penalizes Both Players (Fully Cooperative)
  - Agents need to learn to keep the ball in the game for as long as possible.
  - Penalizing both of the players whenever the ball goes out of play.

	Left player scores	Right player scores
Left player reward	-1	-1
Right player reward	-1	-1

Tampuu, Ardi, et al. "Multiagent cooperation and competition with deep reinforcement learning." *PloS one* 12.4 (2017): e0172395.

Multi-Agent RL

IQL

- Transition Between Cooperation and Competition
  - The fully competitive and fully cooperative cases both penalize loosing the ball equally. What differentiates the two strategies are the values on the main diagonal of the reward matrix.
  - Allow this reward value ρ to change gradually from -1 to +1

	Left player scores	Right player scores
Left player reward	ρ	-1
Right player reward	-1	ρ

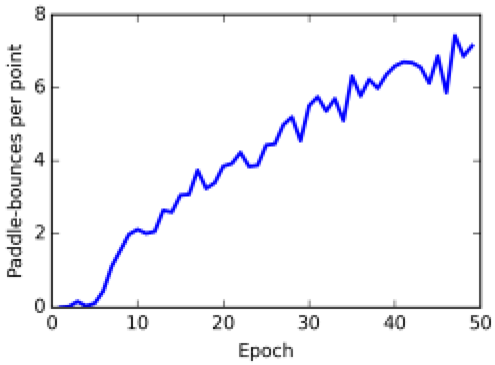
- Three measures
  - Average paddle-bounces per point
  - Average wall-bounces per paddle-bounce
  - Average serving time per point

Tampuu, Ardi, et al. "Multiagent cooperation and competition with deep reinforcement learning." *PloS one* 12.4 (2017): e0172395.

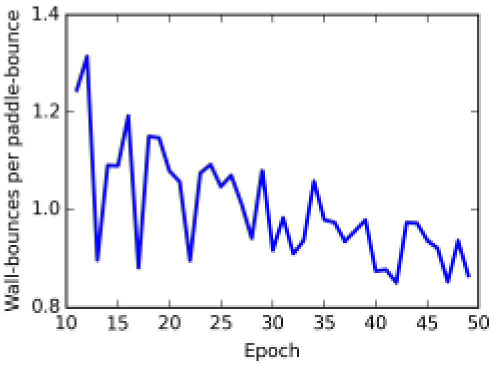
Multi-Agent RL

IQL

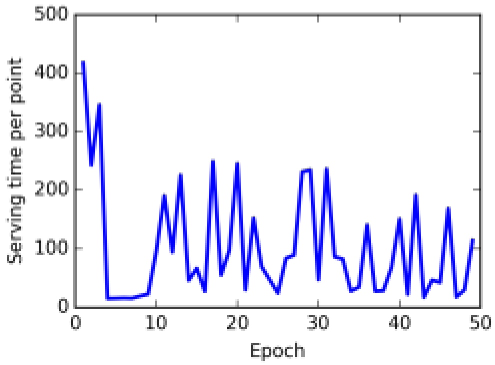
- Result—Fully Competitive



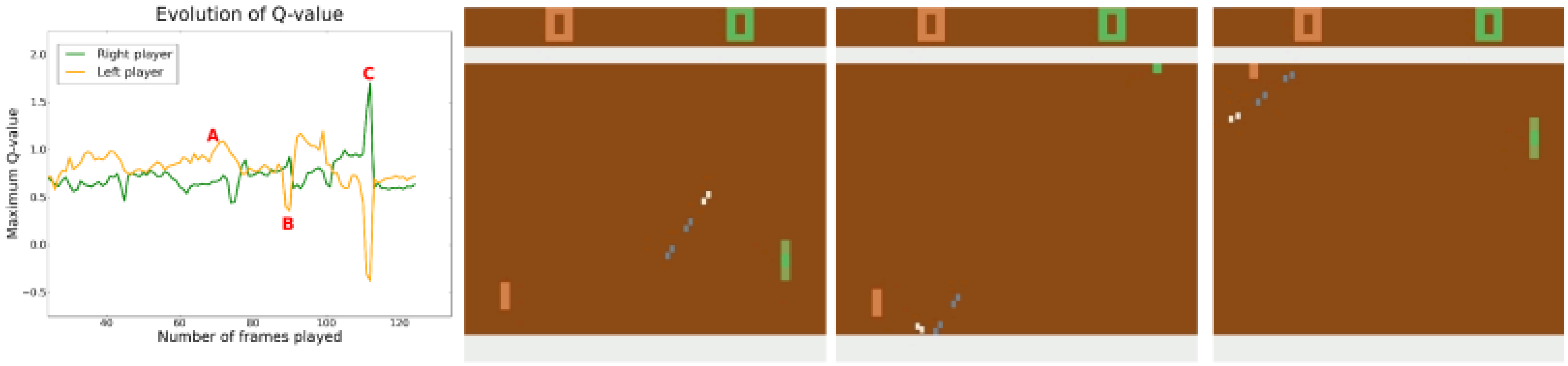
(a) Paddle-bounces per point



(b) Wall-bounces per paddle-bounce



(c) Serving time per point

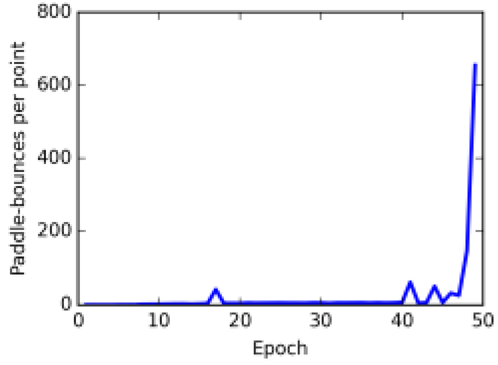


Tampuu, Ardi, et al. "Multiagent cooperation and competition with deep reinforcement learning." *PloS one* 12.4 (2017): e0172395.

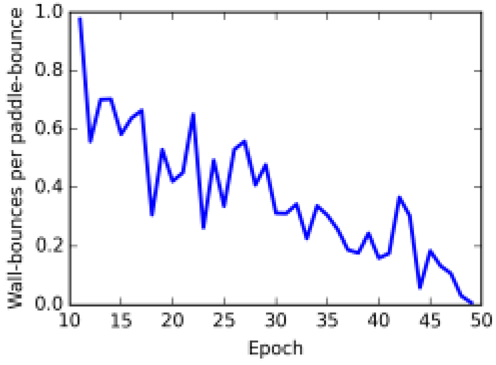
Multi-Agent RL

IQL

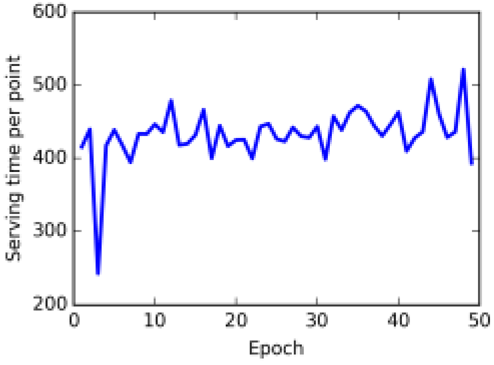
- Result—Fully Cooperative



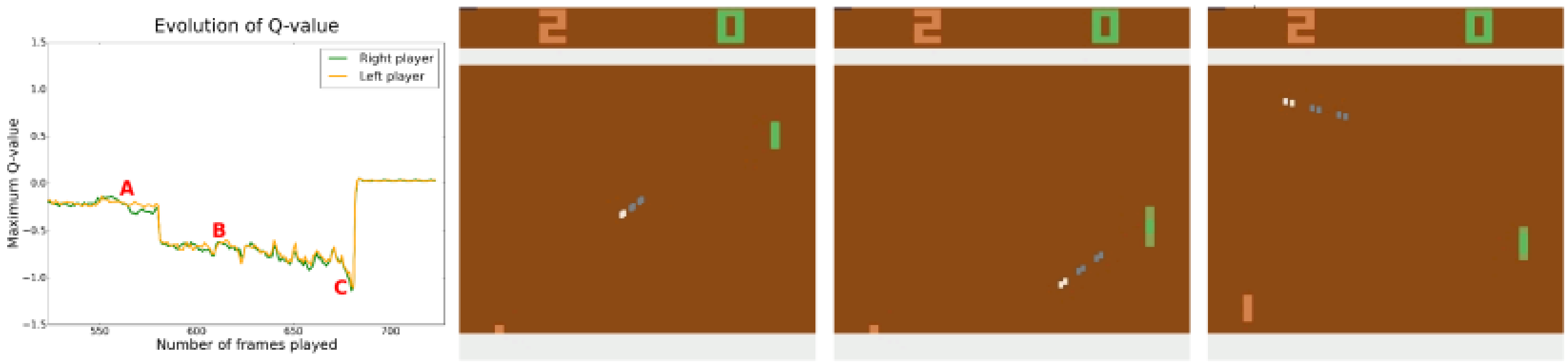
(a) Paddle-bounces per point



(b) Wall-bounces per paddle-bounce



(c) Serving time per point



Tampuu, Ardi, et al. "Multiagent cooperation and competition with deep reinforcement learning." *PloS one* 12.4 (2017): e0172395.

Multi-Agent RL

IQL

- Result—Progression from Competition to Cooperation

Agent	Average paddle-bounces per point	Average wall-bounces per paddle-bounce	Average serving time per point
Competitive $\rho = 1$	7.15 ± 1.01	0.87 ± 0.08	113.87 ± 40.30
Transition $\rho = 0.75$	7.58 ± 0.71	0.83 ± 0.06	129.03 ± 38.81
Transition $\rho = 0.5$	6.93 ± 0.49	0.64 ± 0.03	147.69 ± 41.02
Transition $\rho = 0.25$	4.49 ± 0.43	1.11 ± 0.07	275.90 ± 38.69
Transition $\rho = 0$	4.31 ± 0.25	0.78 ± 0.05	407.64 ± 100.79
Transition $\rho = -0.25$	5.21 ± 0.36	0.60 ± 0.05	449.18 ± 99.53
Transition $\rho = -0.5$	6.20 ± 0.20	0.38 ± 0.04	433.39 ± 98.77
Transition $\rho = -0.75$	409.50 ± 535.24	0.02 ± 0.01	591.62 ± 302.15
Cooperative $\rho = -1$	654.66 ± 542.67	0.01 ± 0.00	393.34 ± 138.63

Tampuu, Ardi, et al. "Multiagent cooperation and competition with deep reinforcement learning." *PloS one* 12.4 (2017): e0172395.

Multi-Agent RL

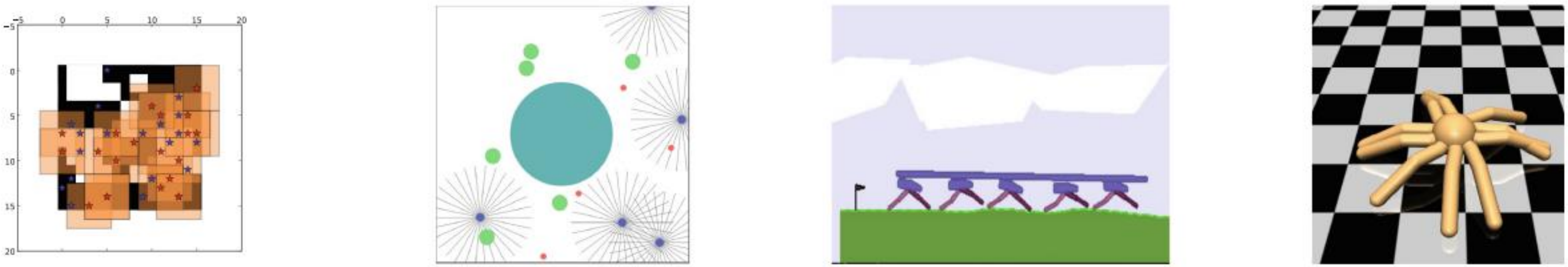
- IL with parameter sharing
  - Dec-POMDPs.
  - Based on DQN, TRPO, DDPG, A3C
  - Three training schemes:
    - Centralized training and execution: a centralized policy maps the joint observation of all agents to a joint action
    - Concurrent training with decentralized execution: each agent learns its own individual policy.
    - Parameter sharing during training with decentralized execution: it allows the policy to be trained with the experiences of all agents simultaneously.

Gupta, Jayesh K., Maxim Egorov, and Mykel Kochenderfer. "Cooperative multi-agent control using deep reinforcement learning." *International conference on autonomous agents and multiagent systems*. Springer, Cham, 2017.

Multi-Agent RL

IL with parameter sharing

- Four multi-agent benchmark tasks
  - Discrete: Pursuit
  - Continuous: Waterworld，Multi-Walker，Multi-Ant



(a) Pursuit (b) Waterworld (c) Multi-Walker (d) Multi-Ant

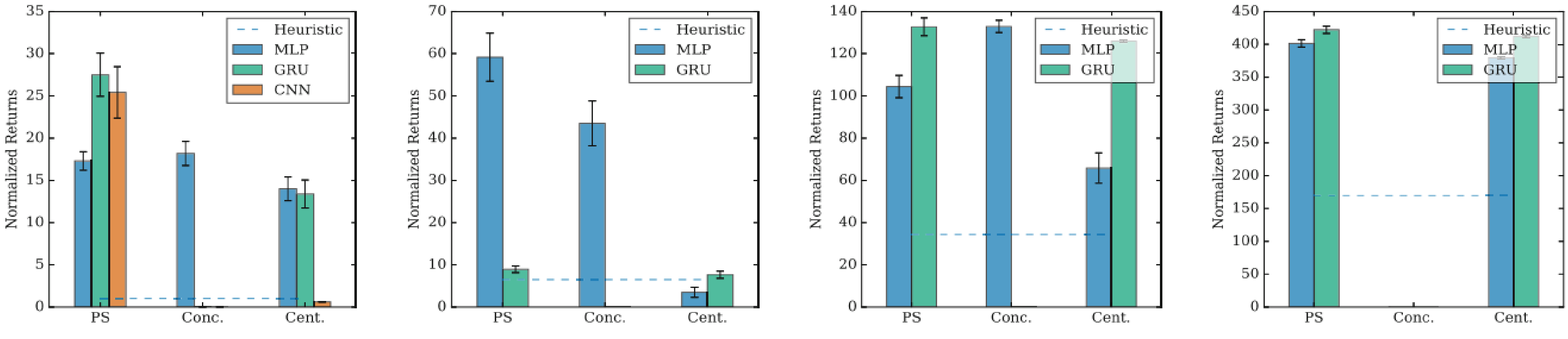
Gupta, Jayesh K., Maxim Egorov, and Mykel Kochenderfer. "Cooperative multi-agent control using deep reinforcement learning." *International conference on autonomous agents and multiagent systems*. Springer, Cham, 2017.

Multi-Agent RL

IL with parameter sharing

- Result

	TRPO	DDPG/DQN	A3C
Feature Net	100-50-25	400-300	128
Recurrent	GRU-32	NA	LSTM-128
Activation	tanh	ReLU	tanh



(a) Pursuit (b) Waterworld (c) Multi-Walker (d) Multi-Ant

Gupta, Jayesh K., Maxim Egorov, and Mykel Kochenderfer. "Cooperative multi-agent control using deep reinforcement learning." *International conference on autonomous agents and multiagent systems*. Springer, Cham, 2017.

Multi-Agent RL

- IL with parameter sharing
  - Result

Task	PS-DQN/DDPG	PS-A3C	PS-TRPO
Pursuit	10.1 ± 6.3	25.5 ± 5.4	17.4 ± 4.9
Waterworld	NA	10.1 ± 5.7	49.1 ± 5.7
Multiwalker	-8.3 ± 3.2	12.4 ± 6.1	58.0 ± 4.2
Multi-ant	307.2 ± 13.8	483.4 ± 3.4	488.1 ± 1.3

Gupta, Jayesh K., Maxim Egorov, and Mykel Kochenderfer. "Cooperative multi-agent control using deep reinforcement learning." *International conference on autonomous agents and multiagent systems*. Springer, Cham, 2017.

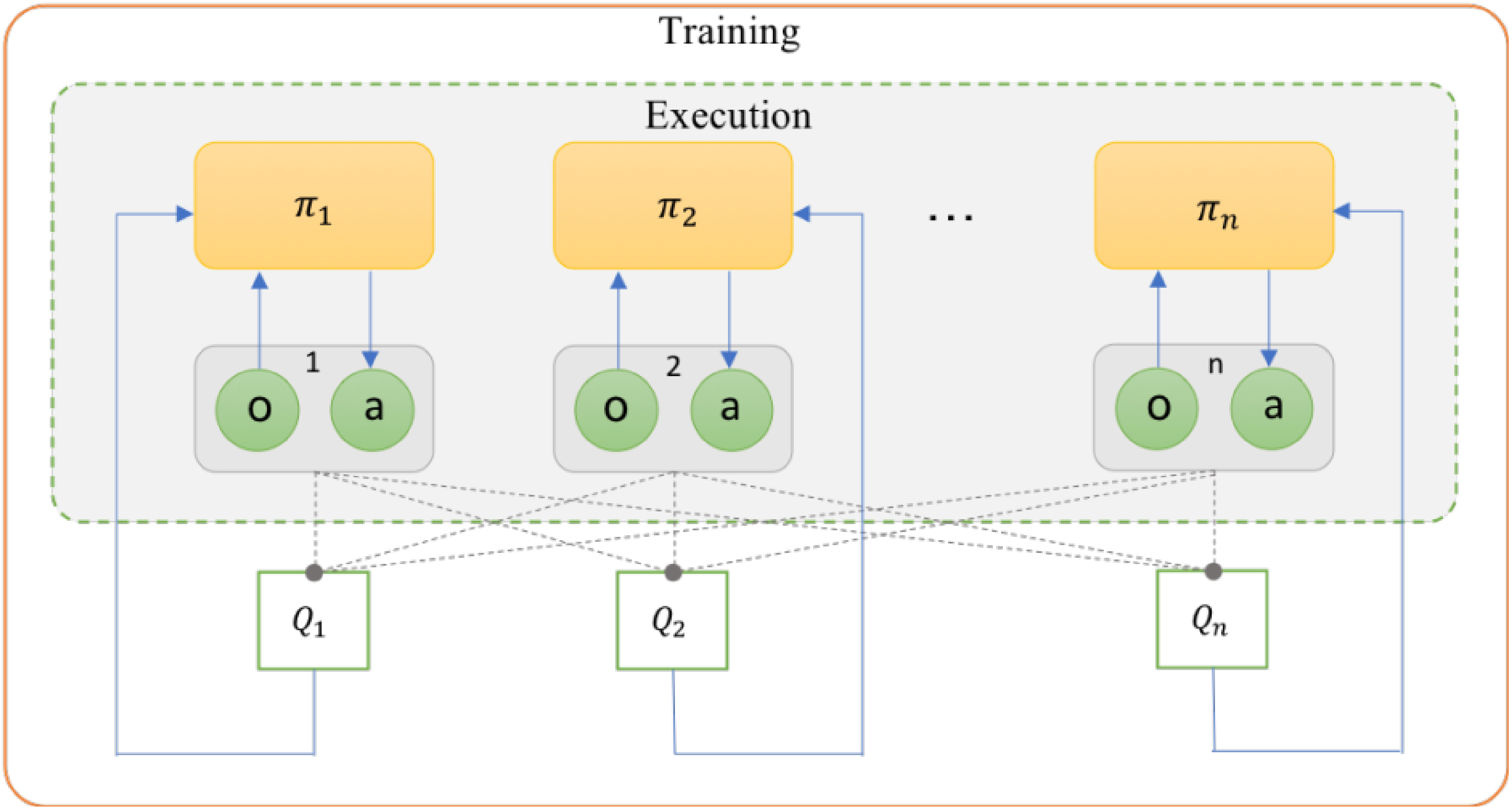
Multi-Agent RL

- Independent Learning
  - IQL
  - IL with parameter sharing

- Learning cooperation
  - MADDPG
  - COMA
  - VDN
  - QMIX

Multi-Agent RL

Centralized Train and Decentralized execution(CTDE)



Multi-Agent RL

MADDPG

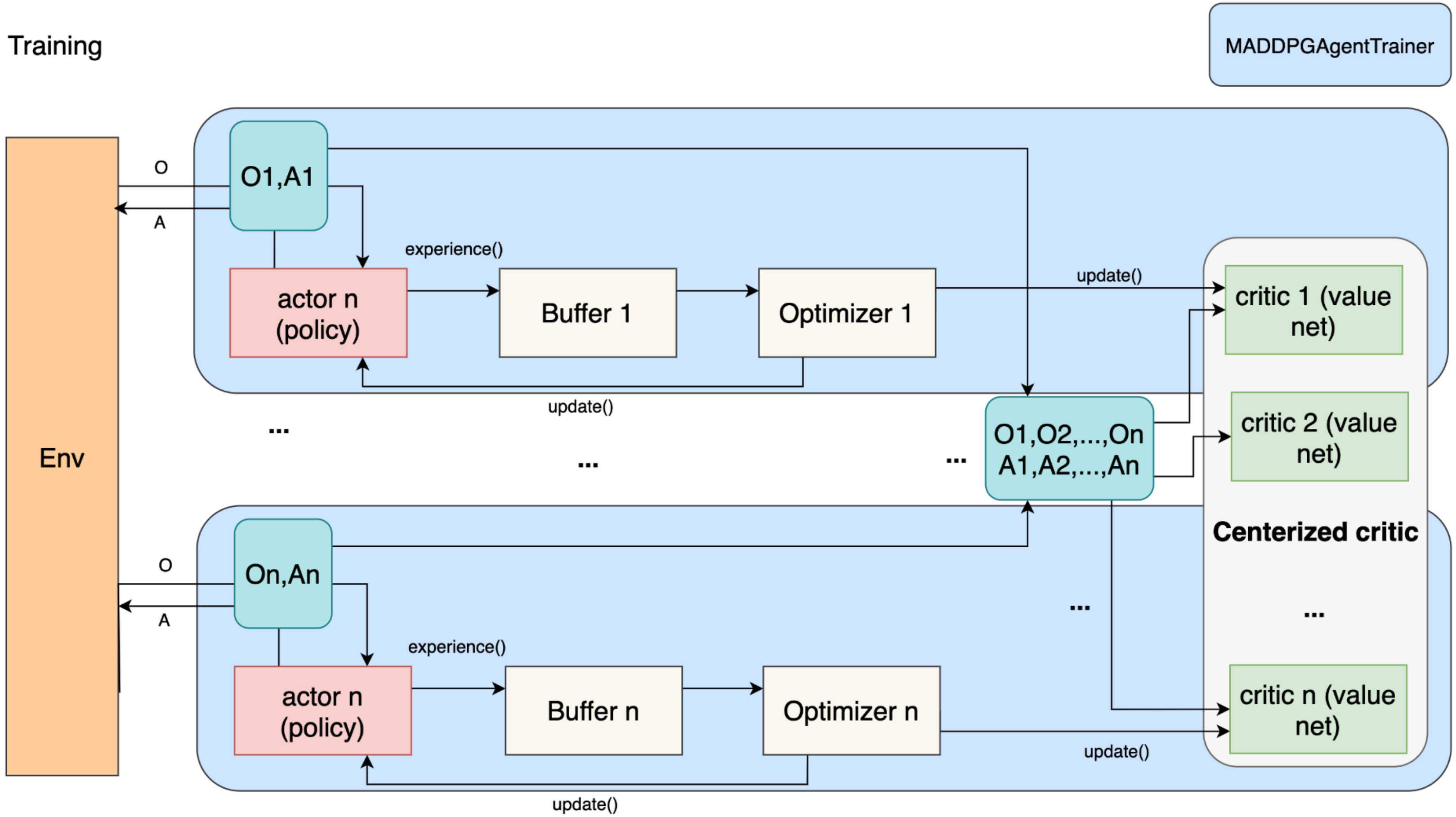
- Challenge
  - Non-stationarity of the environment
  - Policy gradient suffers from a variance that increases as the number of agents grows
- Main idea
  - Leads to learned policies that only use local information at execution time and allow the policies to use extra information to ease training
  - Does not assume a differentiable model of the environment dynamics or any particular structure on the communication method between agents
  - Is applicable not only to cooperative interaction but to competitive or mixed interaction involving both physical and communicative behavior

Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." *Advances in neural information processing systems* 30 (2017).

Multi-Agent RL

MADDPG

- Simple extension of actor-critic policy gradient methods where critic is augmented with extra information about the policies of other agents, while the actor only has access to local information



Multi-Agent RL

MADDPG

- The gradient of the expected return for agent i:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s\sim p^{\boldsymbol{\mu}}, a_i\sim \boldsymbol{\pi}_i}\left[\nabla_{\theta_i} \log \boldsymbol{\pi}_i(a_i \mid o_i) Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, \ldots, a_N)\right], \mathbf{x} = (o_1, \ldots, o_n)$$

- Deterministic policies:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x}, a\sim \mathcal{D}}\left[\nabla_{\theta_i} \boldsymbol{\mu}_i(a_i | \ o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N)\Big|_{a_i=\boldsymbol{\mu}_i(o_i)}\right]$$

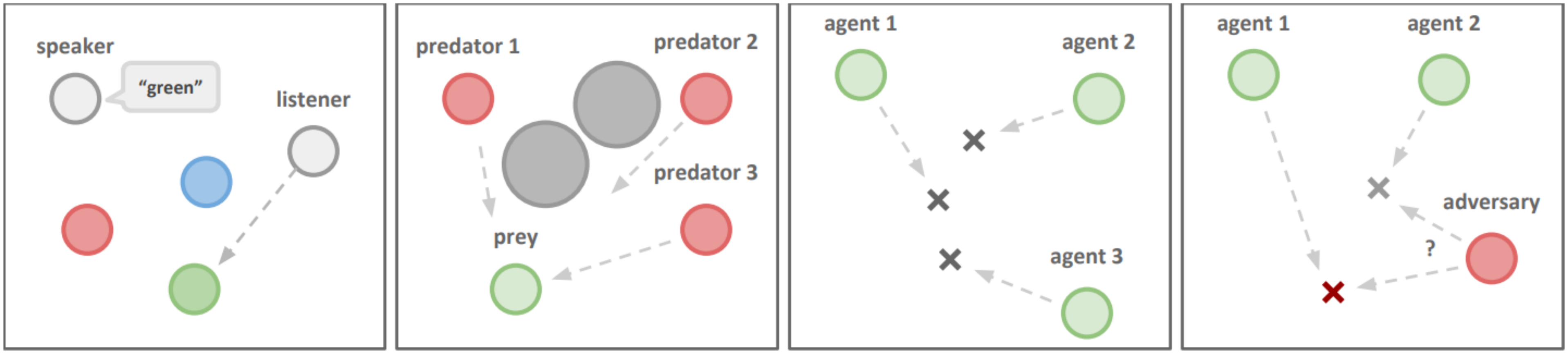
$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x}, a, r, \mathbf{x}'}\left[(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y)^2\right], y = r_i + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a_1', \ldots, a_N') \mid a_j' = \boldsymbol{\mu}_j'(o_j)$$

Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." *Advances in neural information processing systems* 30 (2017).

Multi-Agent RL

MADDPG

- Multiagent-particle-environment
  - Cooperative Communication
  - Predator-Prey
  - Cooperative Navigation
  - Physical Deception
- The environments are publicly available: https://github.com/openai/multiagent-particle-envs

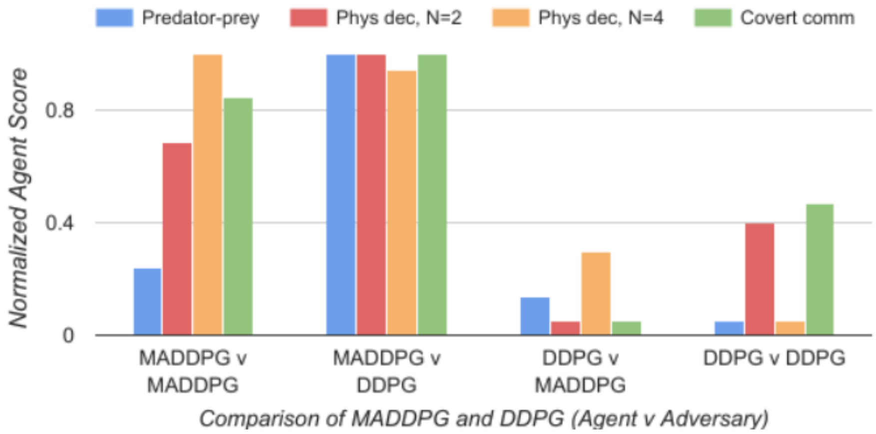


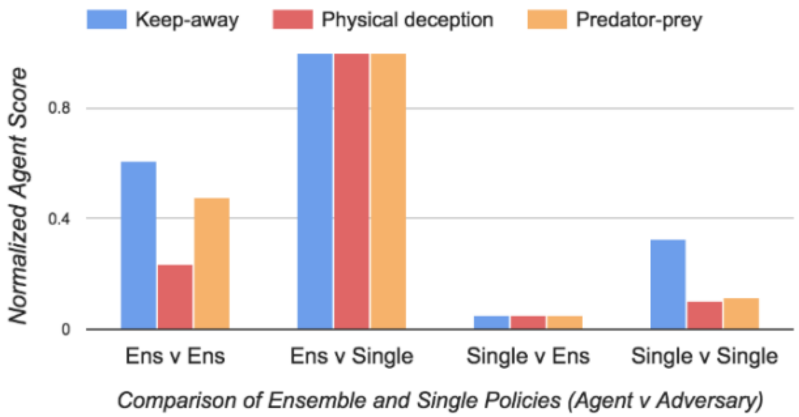
Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." *Advances in neural information processing systems* 30 (2017).

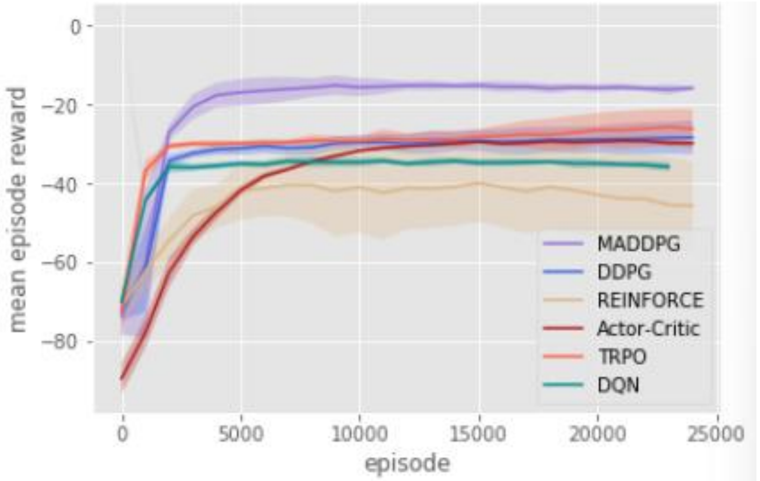
Multi-Agent RL

MADDPG

- Performance







Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." *Advances in neural information processing systems* 30 (2017).

COMA

- Challenge
  - Credit assignment: in cooperative settings, joint actions typically generate only global rewards, making it difficult for each agent to deduce its own contribution to the team's success.
- Main idea
  - Use CTDE as same as MADDPG
  - Counterfactual: Use a counterfactual baseline that marginalizes out a single agent's action
    - Using the centralized critic to compute an agent-specific advantage function that compares the estimated return for the current joint action to a counterfactual baseline that marginalizes out a single agent's action, while keeping the other agent's actions fixed
  - Uses a critic representation that allows the counterfactual baseline to be computed efficiently.

Foerster, Jakob, et al. "Counterfactual multi-agent policy gradients." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. No. 1. 2018.

COMA

- Advantage function:

$$A^a(s,\mathbf{u}) = Q(s,\mathbf{u}) - \sum \pi^a\left(u'^a \mid \tau^a\right) Q\left(s,\left(\mathbf{u}^{-a}, u'^a\right)\right)$$

- COMA gradient:

$$g = \mathbb{E}_\pi\left[\sum_a \nabla_\theta log \pi^a(u^a \mid \tau^a) A^a(s,\mathbf{u})\right], A^a(s,\mathbf{u}) = Q(s,\mathbf{u}) - b(s,\mathbf{u}^{-a})$$

$$g_b = -\mathbb{E}_\pi\left[\sum_a \nabla_\theta log \pi^a(u^a \mid \tau^a) b(s,\mathbf{u}^{-a})\right]$$

$$\begin{aligned} g_b &= -\sum_s d^\pi(s) \sum_a \sum_{\mathbf{u}^{-a}} \pi\left(\mathbf{u}^{-a} \mid \tau^{-a}\right) \sum_{u^a} \pi^a\left(u^a \mid \tau^a\right) \nabla_\theta \log \pi^a\left(u^a \mid \tau^a\right) b\left(s,\mathbf{u}^{-a}\right) \\ &= -\sum_s d^\pi(s) \sum_a \sum_{\mathbf{u}^{-a}} \pi\left(\mathbf{u}^{-a} \mid \tau^{-a}\right) \sum_{u^a} \nabla_\theta \pi^a\left(u^a \mid \tau^a\right) b\left(s,\mathbf{u}^{-a}\right) \\ &= -\sum_s d^\pi(s) \sum_a \sum_{\mathbf{u}^{-a}} \pi\left(\mathbf{u}^{-a} \mid \tau^{-a}\right) b\left(s,\mathbf{u}^{-a}\right) \nabla_\theta 1 \\ &= 0 \end{aligned}$$

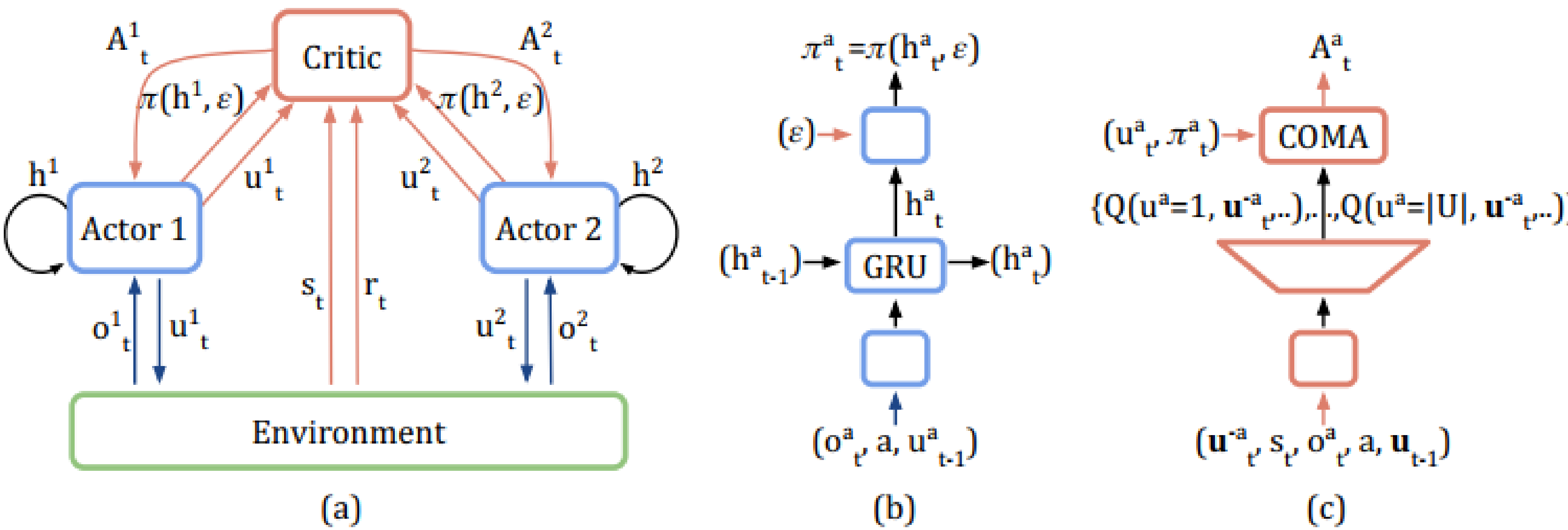
Foerster, Jakob, et al. "Counterfactual multi-agent policy gradients." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. No. 1. 2018.

COMA

- COMA gradient：

$$g = \mathbb{E}_{\pi}\left[\sum_{a} \nabla_{\theta} \log \pi^{a}(u^{a} \mid \tau^{a}) A^{a}(s,\mathbf{u})\right], A^{a}(s,\mathbf{u}) = Q(s,\mathbf{u}) - b(s,\mathbf{u}^{-a})$$

$$= \mathbb{E}_{\pi}\left[\sum_{a} \nabla_{\theta} \log \pi^{a}(u^{a} \mid \tau^{a}) Q(s,\mathbf{u})\right] = \mathbb{E}_{\pi}\left[\nabla_{\theta} \log \prod_{a} \pi^{a}(u^{a} \mid \tau^{a}) Q(s,\mathbf{u})\right]$$



Foerster, Jakob, et al. "Counterfactual multi-agent policy gradients." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. No. 1. 2018.

COMA

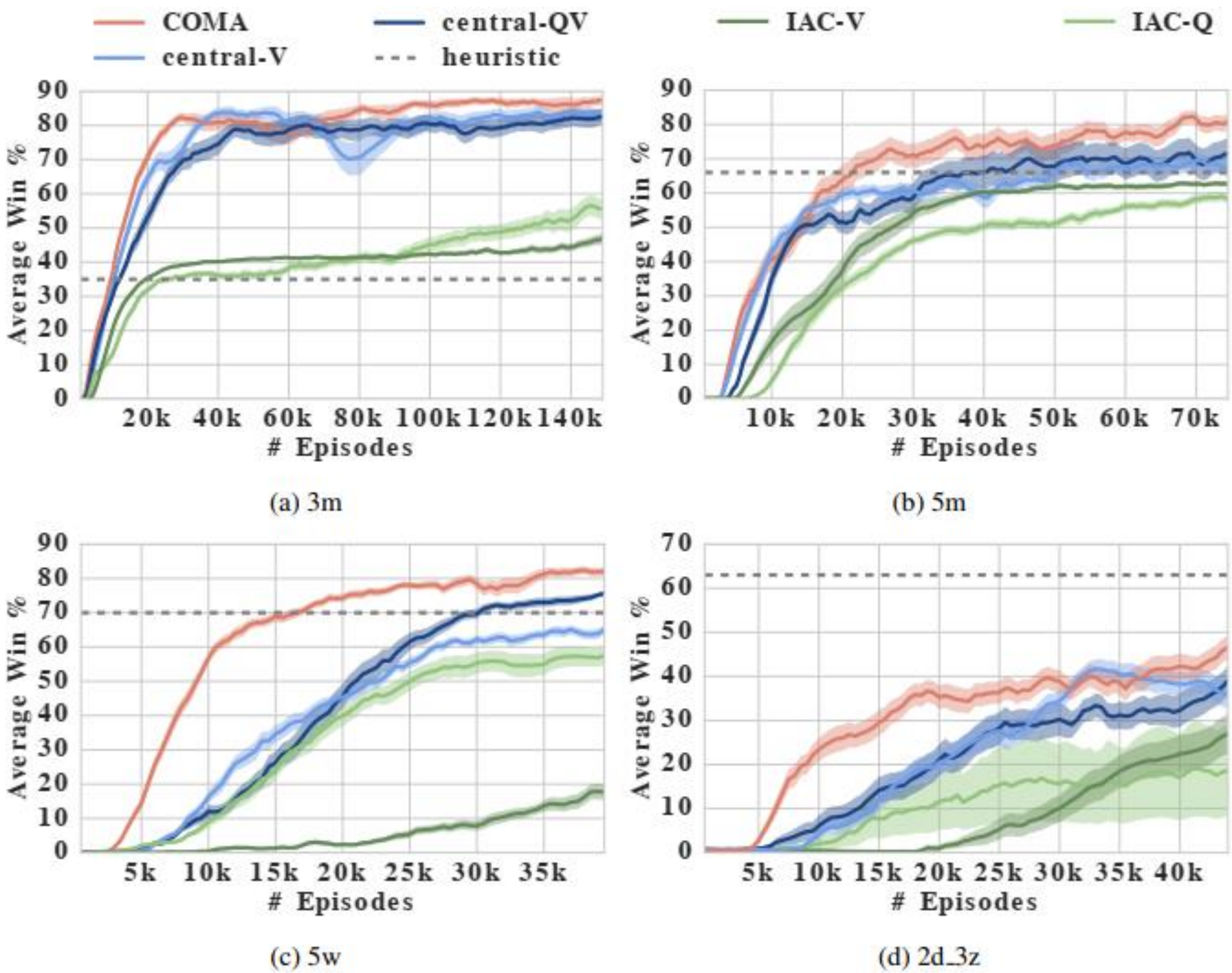
Algorithm 1 Counterfactual Multi-Agent (COMA) Policy Gradients

Initialise $\theta_1^c, \hat{\theta}_1^c, \theta^\pi$
for each training episode e **do**
  Empty buffer
  for $e_c = 1$ **to** $\frac{\text{BatchSize}}{n}$ **do**
    s_1 = initial state, $t = 0$, $h_0^a = \mathbf{0}$ for each agent a
    while $s_t \neq$ terminal **and** $t < T$ **do**
      $t = t + 1$
      for each agent a **do**
        $h_t^a = \text{Actor}\left(o_t^a, h_{t-1}^a, u_{t-1}^a, a, u; \theta_i\right)$
        Sample u_t^a from $\pi(h_t^a, \epsilon(e))$
      Get reward r_t and next state s_{t+1}
    Add episode to buffer
  Collate episodes in buffer into single batch
  for $t = 1$ **to** T **do** // from now processing all agents in parallel via single batch
    Batch unroll RNN using states, actions and rewards
    Calculate TD(λ) targets y_t^a using $\hat{\theta}_i^c$
  for $t = T$ **down to** 1 **do**
    $\Delta Q_t^a = y_t^a - Q\left(s_j^a, \mathbf{u}\right)$
    $\Delta\theta^c = \nabla_{\theta^c}(\Delta Q_t^a)^2$ // calculate critic gradient
    $\theta_{i+1}^c = \theta_i^c - \alpha\Delta\theta^c$ // update critic weights
    Every C steps reset $\hat{\theta}_i^c = \theta_i^c$
  for $t = T$ **down to** 1 **do**
    $A^a(s_t^a, \mathbf{u}) = Q(s_t^a, \mathbf{u}) - \sum_u Q(s_t^a, u, \mathbf{u}^{-a})\pi(u|h_t^a)$ // calculate COMA
    $\Delta\theta^\pi = \Delta\theta^\pi + \nabla_{\theta^\pi} \log \pi(u|h_t^a) A^a(s_t^a, \mathbf{u})$ // accumulate actor gradients
  $\theta_{i+1}^\pi = \theta_i^\pi + \alpha\Delta\theta^\pi$ // update actor weights

Foerster, Jakob, et al. "Counterfactual multi-agent policy gradients." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. No. 1. 2018.

Multi-Agent RL

- **COMA**
  - Performance



Foerster, Jakob, et al. "Counterfactual multi-agent policy gradients." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. No. 1. 2018.

Multi-Agent RL

VDN

- **Challenge**
  - Lazy agent: one agent learns a useful policy, but a second agent is discouraged from learning because its exploration would hinder the first agent and lead to worse team reward.
  - Large-scale multi-agent scenarios.
- **Main idea**
  - Training individual agents with a novel value decomposition network architecture, which learns to decompose the team value function into agent-wise value functions
  - The value decomposition network aims to learn an optimal linear value decomposition from the team reward signal, by back-propagating the total Q gradient through deep neural networks representing the individual component value functions.

Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." *arXiv preprint arXiv:1706.05296* (2017).

Multi-Agent RL

VDN

- Joint action-value function can be additively decomposed into value functions across agents

$$Q((h^1,\ldots,h^d),(a^1,\ldots,a^d)) \approx \sum_{i=1}^{d} \tilde{Q}_i(h^i,a^i)$$

- Consider the case with two agents and where rewards decompose additively across agent observations, $r(\mathbf{s},\mathbf{a}) = r_1\left(o_t^1, a_t^1\right) + r_2\left(o_t^2, a_t^2\right)$

$$\begin{aligned} Q^{\pi}(\mathbf{s},\mathbf{a}) \quad &= \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r\left(\mathbf{s}_t,\mathbf{a}_t\right) \mid \mathbf{s}_1 = \mathbf{s}, \mathbf{a}_1 = \mathbf{a}; \pi\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_1\left(o_t^1, a_t^1\right) \mid \mathbf{s}_1 = \mathbf{s}, \mathbf{a}_1 = \mathbf{a}; \pi\right] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_2\left(o_t^2, a_t^2\right) \mid \mathbf{s}_1 = \mathbf{s}, \mathbf{a}_1 = \mathbf{a}; \pi\right] \\ &=: \bar{Q}_1^{\pi}(\mathbf{s},\mathbf{a}) + \bar{Q}_2^{\pi}(\mathbf{s},\mathbf{a}) \end{aligned}$$

- When agents store additional information from historical observation

$$Q^{\pi}(\mathbf{s},\mathbf{a}) =: \bar{Q}_1^{\pi}(\mathbf{s},\mathbf{a}) + \bar{Q}_2^{\pi}(\mathbf{s},\mathbf{a}) \approx \tilde{Q}_1^{\pi}(h^1,a^1) + \tilde{Q}_2^{\pi}(h^2,a^2)$$

Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." *arXiv preprint arXiv:1706.05296* (2017).

Multi-Agent RL

VDN

- Architecture

IL

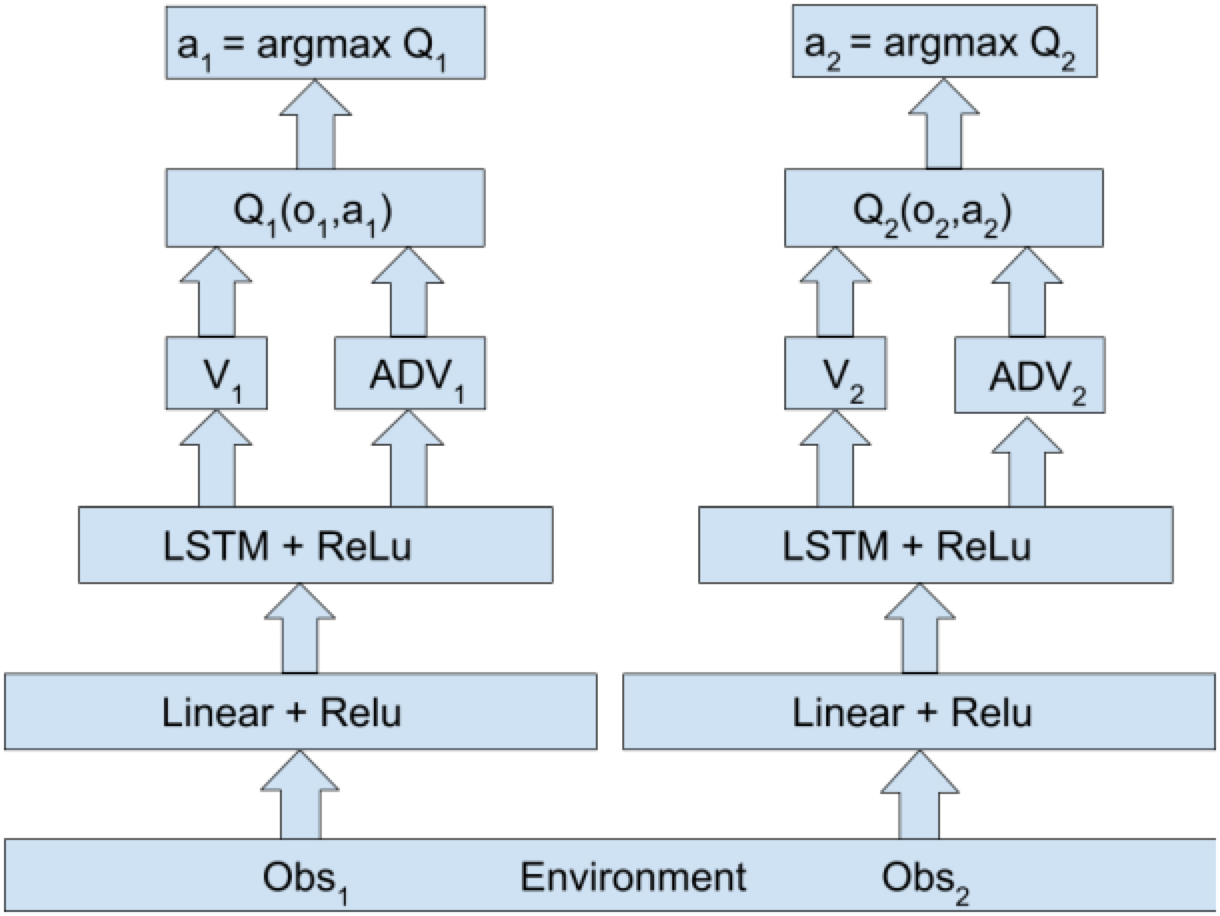


Figure 14: Independent Agents Architecture

Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." *arXiv preprint arXiv:1706.05296* (2017).

Multi-Agent RL

VDN

- Architecture

VDN

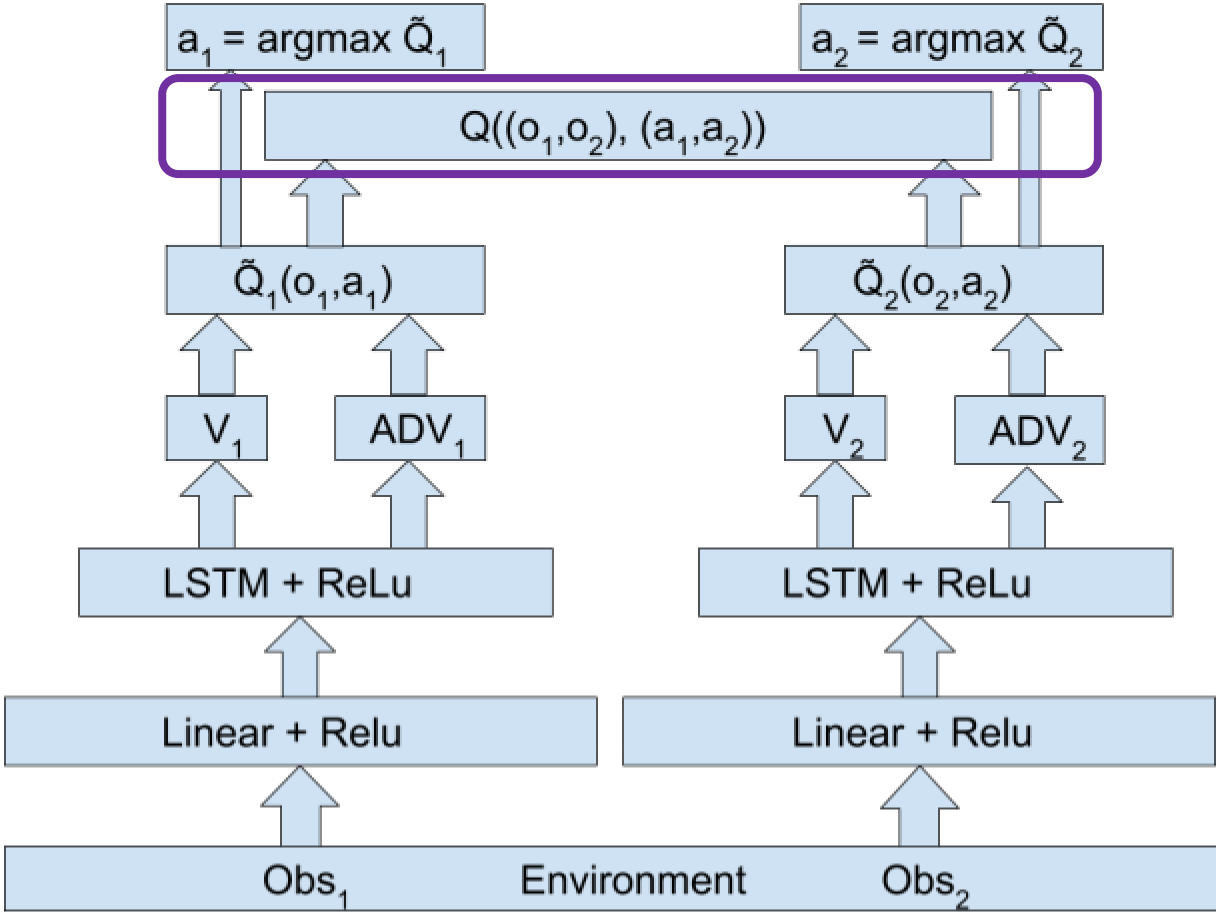


Figure 15: Value-Decomposition Individual Architecture

Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." *arXiv preprint arXiv:1706.05296* (2017).

Multi-Agent RL

VDN

- Architecture

VDN+Comm

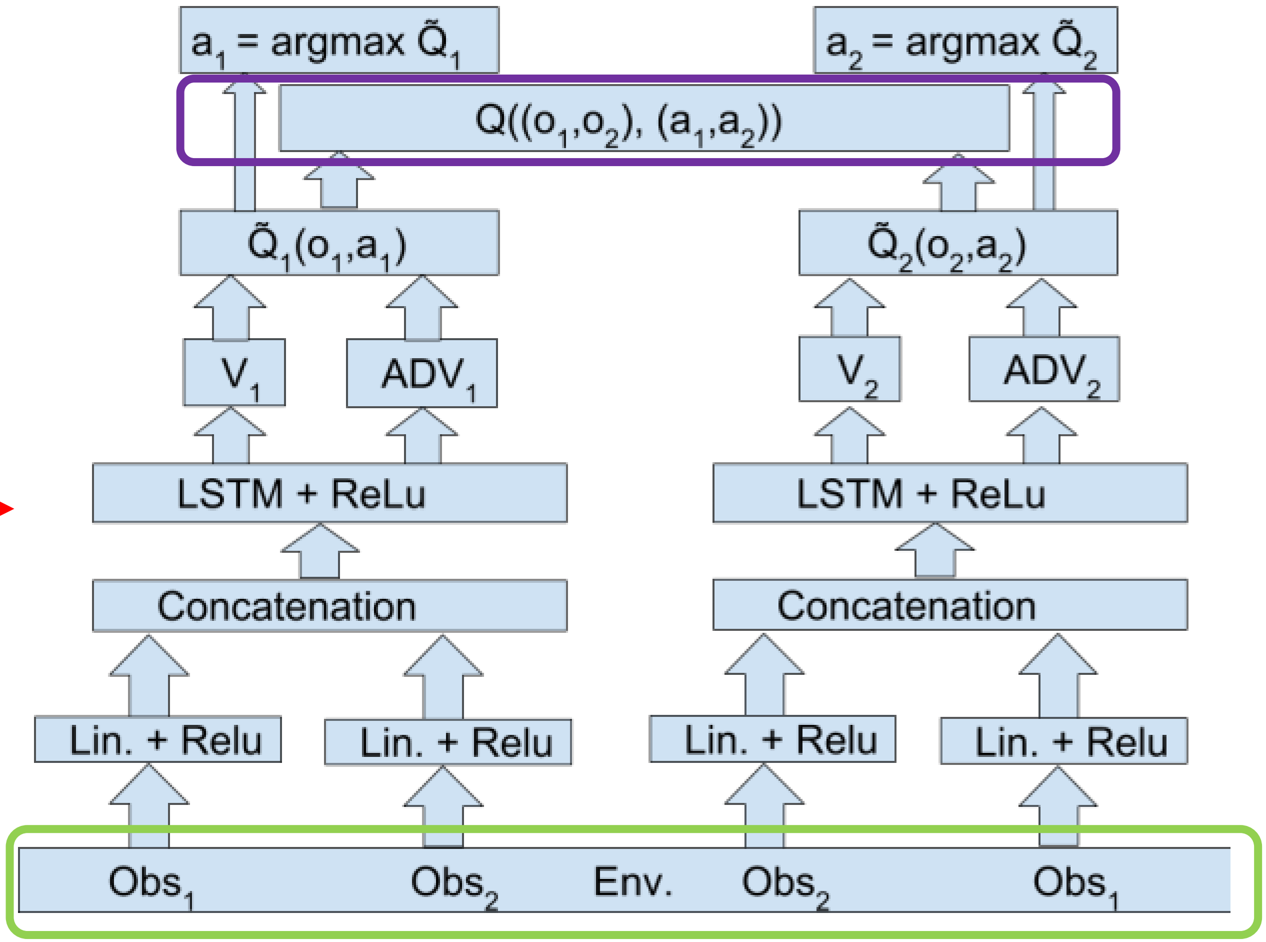


Figure 16: Low-level communication Architecture

Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." *arXiv preprint arXiv:1706.05296* (2017).

Multi-Agent RL

VDN

- Architecture

VDN+Comm

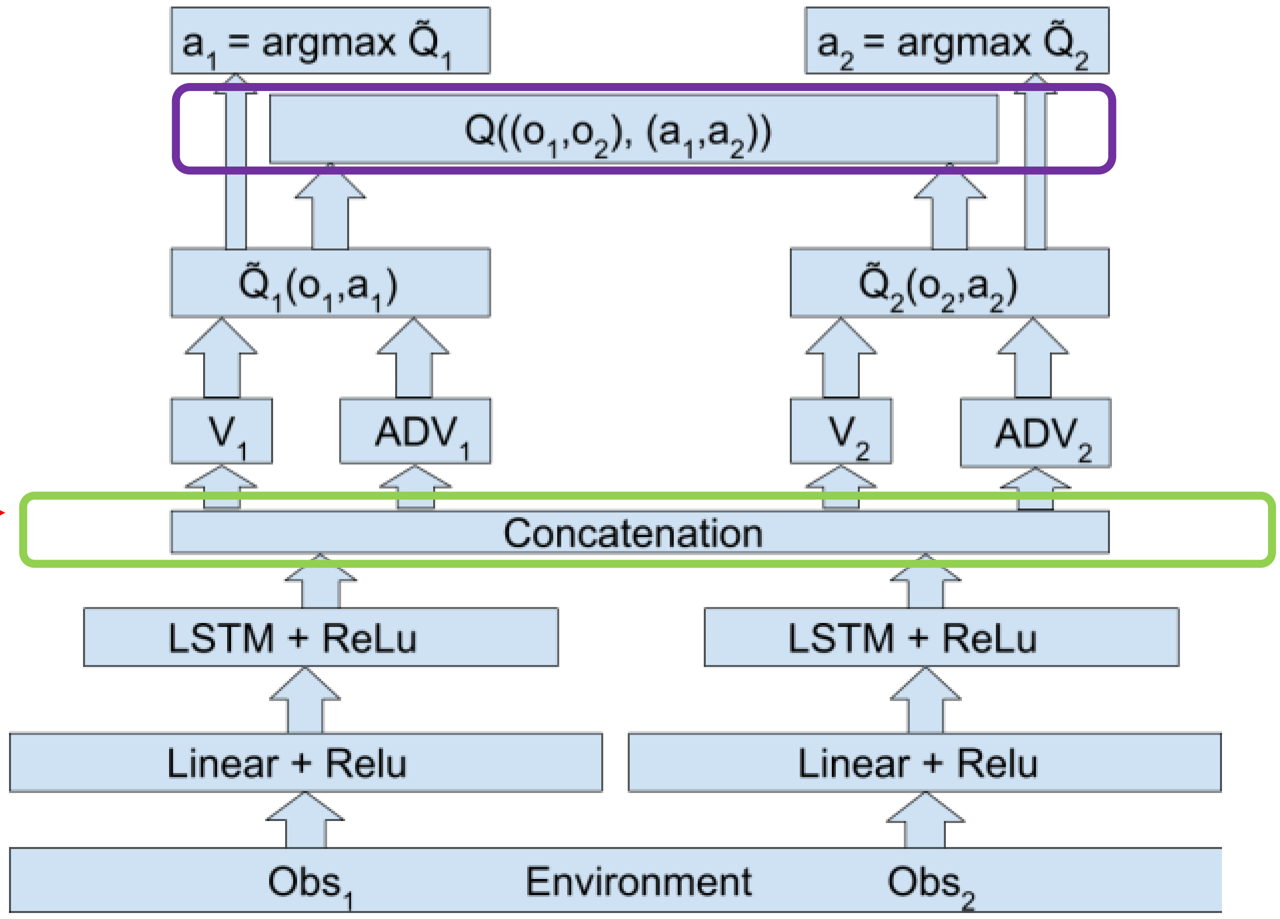


Figure 17: High-level communication Architecture

Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." *arXiv preprint arXiv:1706.05296* (2017).

Multi-Agent RL

VDN

- Architecture

Centralized

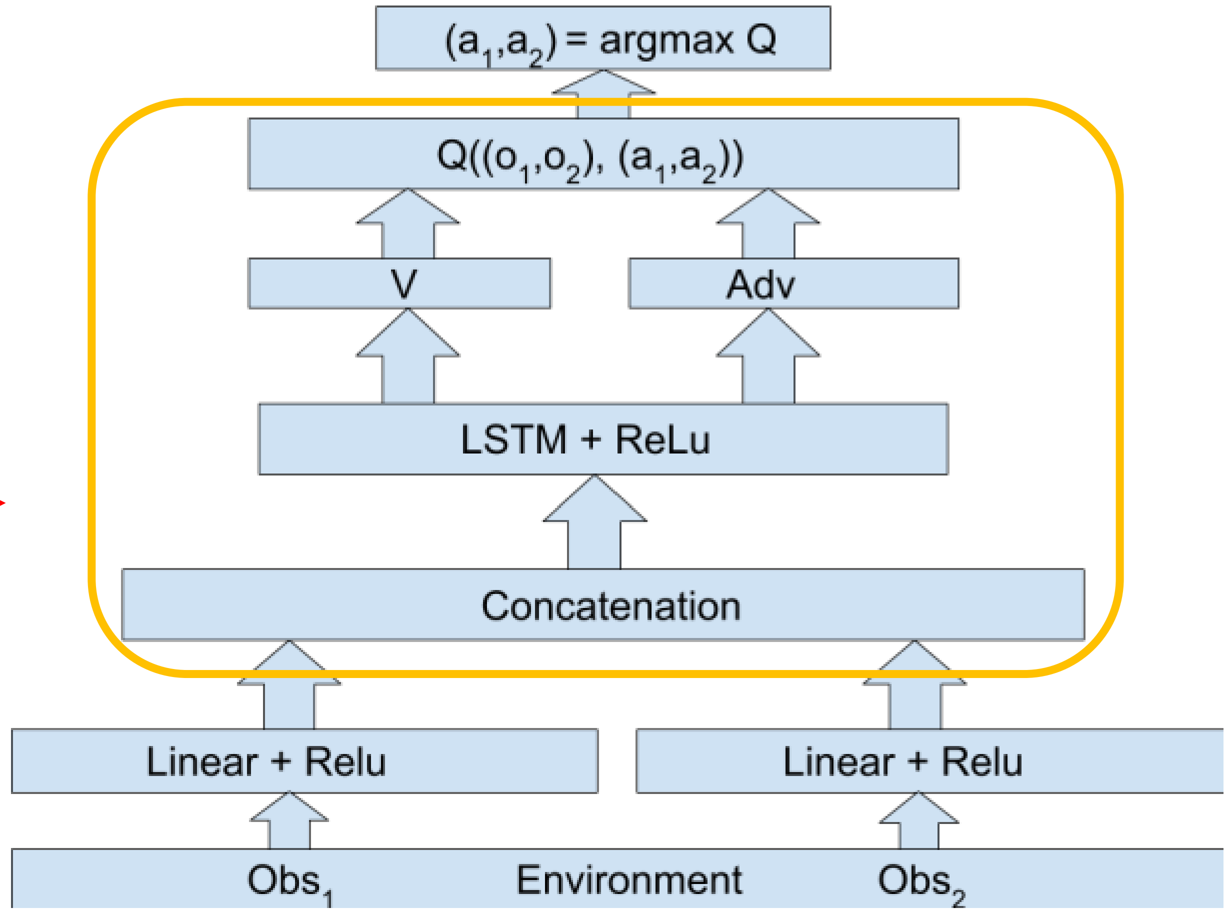


Figure 20: Combinatorially Centralized Architecture

Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." *arXiv preprint arXiv:1706.05296* (2017).

Multi-Agent RL

VDN

- Performance

Agent	V.	S.	Id	L.	H.	C.
1						
2	✓					
3	✓	✓				
4	✓	✓	✓			
5	✓	✓	✓	✓		
6	✓	✓	✓		✓	
7	✓	✓	✓	✓	✓	
8	✓					✓
9						✓

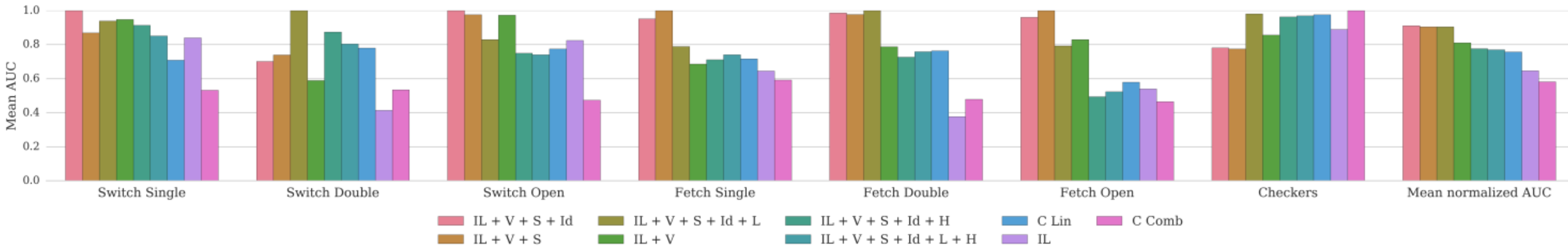
Table 1: Agent architectures. V is value decomposition, S means shared weights and an invariant network, Id means role info was provided, L stands for lower-level communication, H for higher-level communication and C for centralization. These architectures were selected to show the advantages of the independent agent with value-decomposition and to study the benefits of additional enhancements added in a logical sequence.

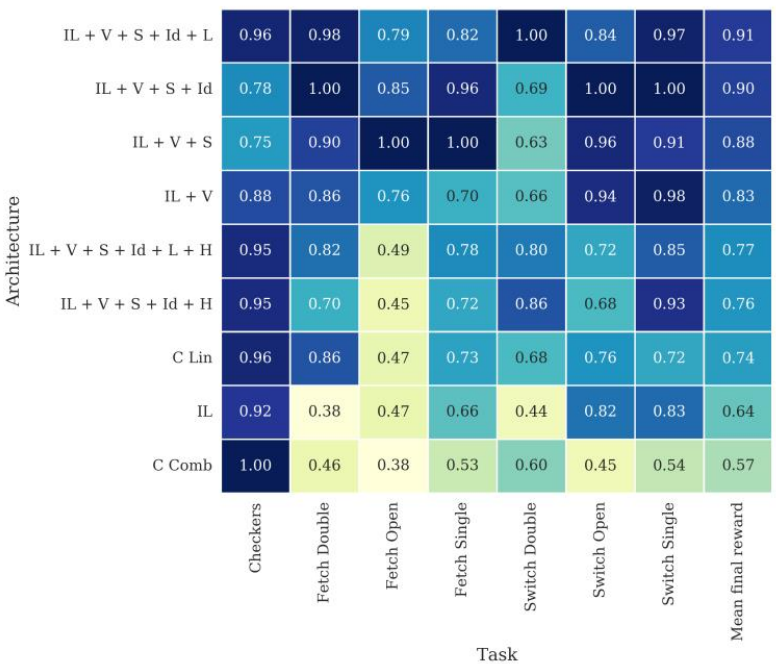
Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." *arXiv preprint arXiv:1706.05296* (2017).

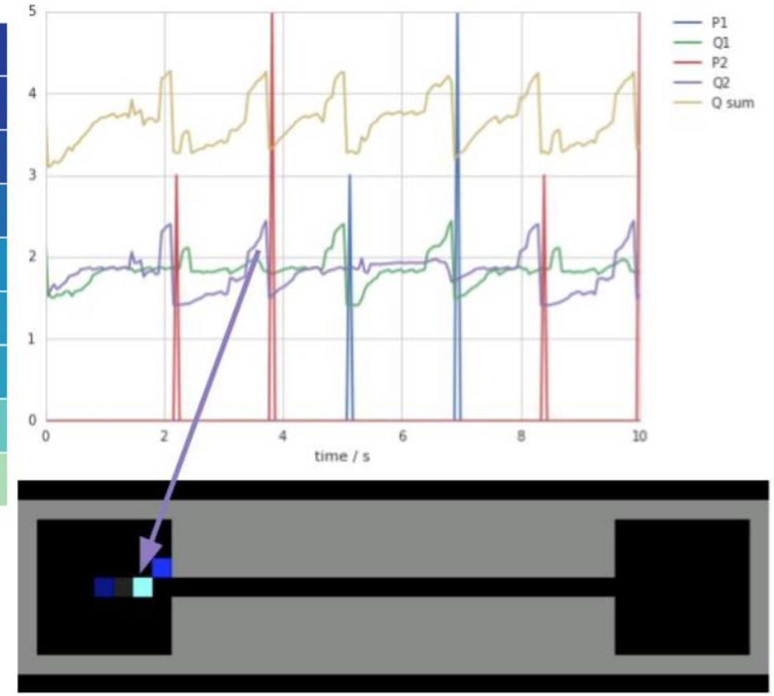
Multi-Agent RL

VDN

- Performance







Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." *arXiv preprint arXiv:1706.05296* (2017).

Multi-Agent RL

QMIX

- Main idea
  - Employ a network that estimates joint action-values as a complex non-linear combination of per-agent values that condition only on local observations
  - Ensure that a global argmax performed on Q_{tot} yields the same result as a set of individual argmax operations performed on each Q_a

$$\underset{\mathbf{u}}{\operatorname{argmax}}\, Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \operatorname{argmax}_{u^1} Q_1\left(\tau^1, u^1\right) \\ \vdots \\ \operatorname{argmax}_{u^n} Q_n\left(\tau^n, u^n\right) \end{pmatrix}$$

  - Enforce a monotonicity constraint on the relationship between Q_{tot} and each Q_a

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A$$

Rashid, Tabish, et al. "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning." *International Conference on Machine Learning*. PMLR, 2018.

Multi-Agent RL

QMIX

- Main idea
  - Enforce a monotonicity constraint on the relationship between Q_{tot} and each Q_a

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A$$

    - Represent Q_{tot} using an architecture consisting of agent networks, a mixing network, and a set of hypernetworks.
    - Restrict the mixing network to have positive weights
    - The weights of the mixing network are produced by separate hypernetworks.
    - Each hypernetwork takes the state s as input and generates the weights of one layer of the mixing network. Each hypernetwork consists of a single linear layer, followed by an absolute activation function, to ensure that the mixing network weights are non-negative.

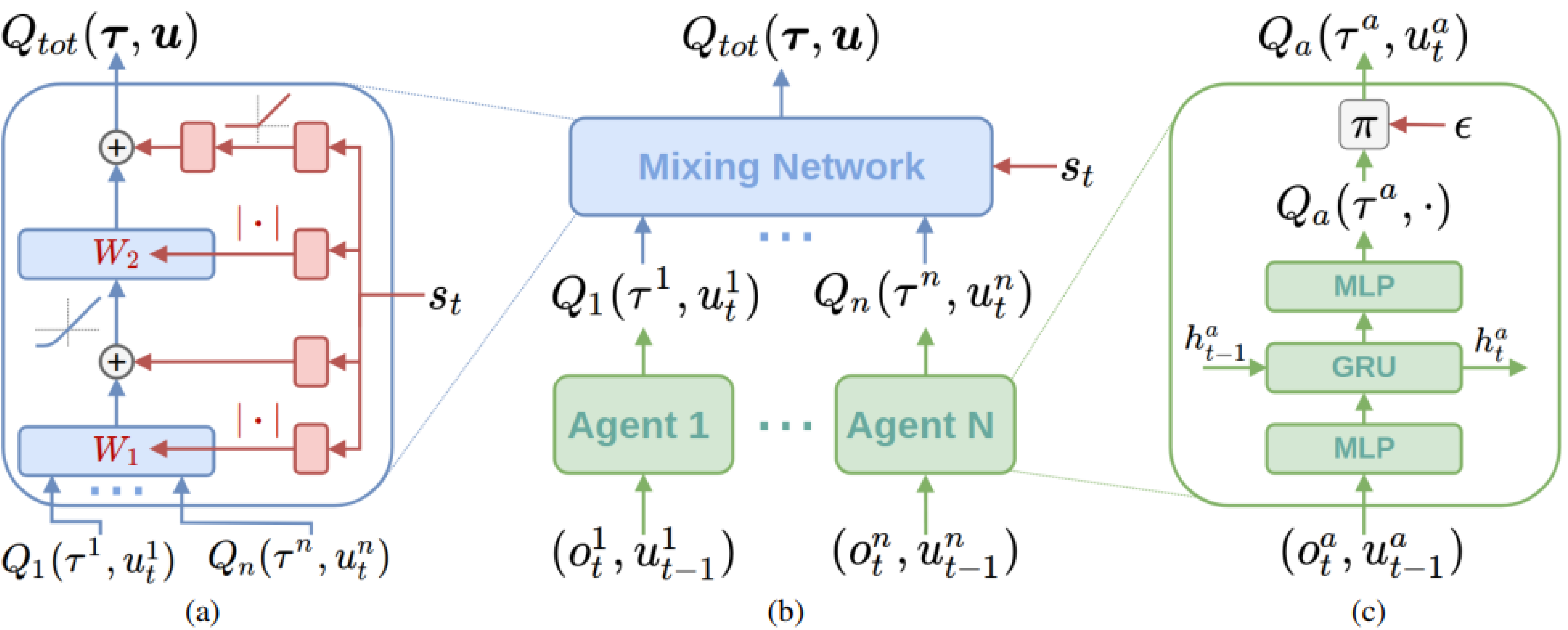
Rashid, Tabish, et al. "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning." *International Conference on Machine Learning*. PMLR, 2018.

Multi-Agent RL

QMIX

- Architecture
- Loss

$$\mathcal{L}(\theta) = \sum_{i=1}^{b}\left[\left(y_i^{tot} - Q_{tot}(\tau, \mathbf{u}, s; \theta)\right)^2\right]$$

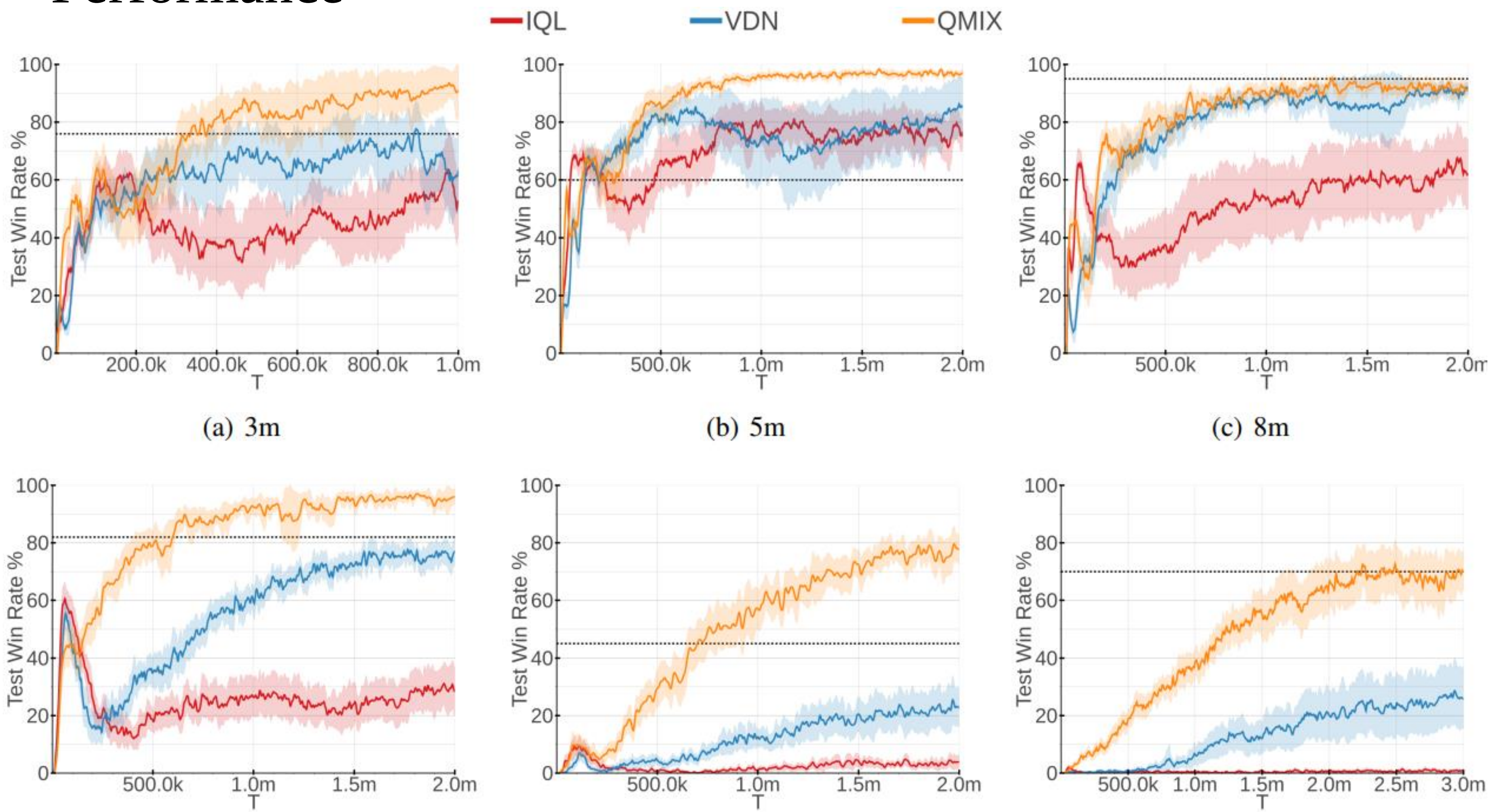


Rashid, Tabish, et al. "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning." *International Conference on Machine Learning*. PMLR, 2018.

Multi-Agent RL

QMIX

➢ Performance



(a) 3m (b) 5m (c) 8m

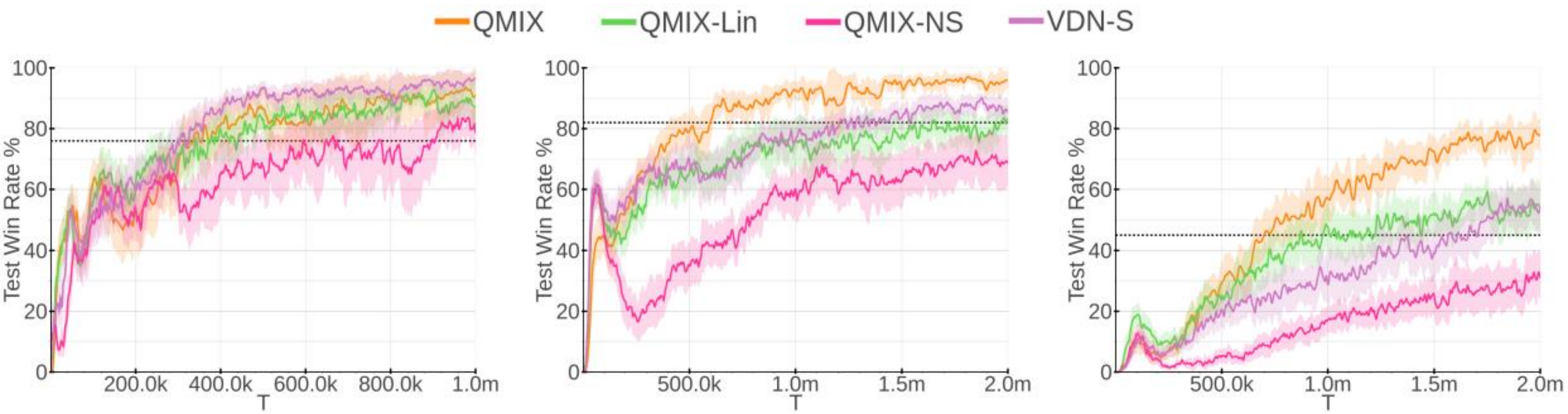
(d) 2s_3z (e) 3s_5z (f) 1c_3s_5z

Rashid, Tabish, et al. "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning." *International Conference on Machine Learning*. PMLR, 2018.

Multi-Agent RL

QMIX

- Performance
  - QMIX-NS: the weights and biases of the mixing network are learned in the standard way, without conditioning on the state
  - QMIX-LIN: remove the hidden layer of the mixing network
  - VDN-S: add a state-dependent term to the sum of the agent's Q-values



(a) 3m (b) 2s_3z (c) 3s_5z

Rashid, Tabish, et al. "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning." *International Conference on Machine Learning*. PMLR, 2018.

Multi-Agent RL

Hyperparameter Tricks

- SOTA multi-agent algorithm
  - QMIX
  - LICA: on-policy, policy mixing network, TD(λ)
  - DOP： off-policy、combines QMIX and COMA、TB(λ)
  - Qatten： attention mechanism
  - QPLEX： decompose Q value into advantages and values(dueling)
  - WQMIX： changing the gradient weight of different samples with a double mixing network architecture

- Hyperparameter trick
  - The hyperparameter trick is an optimization in hyperparameter settings that is unaccounted for in experimental design but which may hold which may hold a significant effect on the outcome

Hu, Jian, et al. "Hyperparameter Tricks in Multi-Agent Reinforcement Learning: An Empirical Study." *arXiv e-prints* (2021): arXiv-2102.

Hyperparameter Tricks

- Hyperparameter trick
  - The hyperparameter trick is an optimization in hyperparameter settings
  - Choice of test Scenarios:
    - some SOTA algorithms select test scenarios that specifically give them starting advantages or avoid the scenarios in which they fail
  - Number of Samples and Number of Rollout Processes
    - S = E * P * I
    - S is the total number of samples，E is the number of samples in each episode，P is the number of rollout processes and I is the number of policy iterations
  - Exploration
    - $\epsilon - greedy$: in some difficult scenarios, we need to perform more steps with high ϵ values

Hu, Jian, et al. "Hyperparameter Tricks in Multi-Agent Reinforcement Learning: An Empirical Study." *arXiv e-prints* (2021): arXiv-2102.

Multi-Agent RL

Hyperparameter Tricks

- Hyperparameter trick
  - Network Size and Optimization
    - Larger neural networks have a higher likelihood to obtain good performance
    - A better neural network optimizer and proper batch size accelerate the convergence of the algorithms
    - Adam
  - Eligibility Traces
    - Returned-based algorithms (where return refers to the sum of discounted rewards $\sum_t \gamma^t r_t$) offer some advantages over value bootstrap algorithms (where return refers to $r_t + V(S_{t+1})$)
    - Eligibility Traces, $Q^*(\lambda)$, TB(λ)
  - Optimal parameters

Hu, Jian, et al. "Hyperparameter Tricks in Multi-Agent Reinforcement Learning: An Empirical Study." *arXiv e-prints* (2021): arXiv-2102.

Hyperparameter Tricks

Comparison

Algorithms	LICA	**Our LICA**	DOP	**Our DOP**
Optimizer	Adam	**Adam**	RMSProp	**Adam**
Batch Size	32	**16**	Off=32, On=16	**Off=64, On=32**
Eligibility traces	TD(λ=0.8)	TD(λ=0.8)	TD(λ=0.8), TB(λ=0.93)	TD(λ=0.8), TB(λ=0.93)
Exploration	Adaptive Entropy=0.06	Adaptive Entropy=0.06	Annealing Noise = 500K	**Adaptive Entropy=0.001**
Critic Net Size (6h_vs_8z)	29696K	29696K	122K	122K
Rollout Processes	32	**16**	4	**8**

Table 2: Hyperparameter changes of Policy-based algorithms

Algorithms	QMIX	**Our QMIX**	Qatten	**Our Qatten**	QPLEX	**Our QPLEX**	WQMIX	**Our WQMIX**
Optimizer	RMSProp	**Adam**	RMSProp	**Adam**	RMSProp	**Adam**	RMSProp	**Adam**
Batch Size	32	**128**	32	**128**	32	**128**	32	**128**
Q(λ)	0	**0.6**	0	**0.6**	0	**0.6**	0	**0.6**
Attention Heads	-	-	4	4	10	**4**	-	-
Mixing Net Size (6h_vs_8z)	41K	41K	58K	58K	476K	**152K**	247K	247K
ε Anneal Steps	50K $\rightarrow$ **500K for** 6h_vs_8z**, 100 K for others**							
Rollout Processes	1	**8**	1	**8**	1	**8**	1	**8**

Table 3: Hyperparameter changes of Value-based algorithms

Hu, Jian, et al. "Hyperparameter Tricks in Multi-Agent Reinforcement Learning: An Empirical Study." *arXiv e-prints* (2021): arXiv-2102.

Multi-Agent RL

Hyperparameter Tricks

- Result

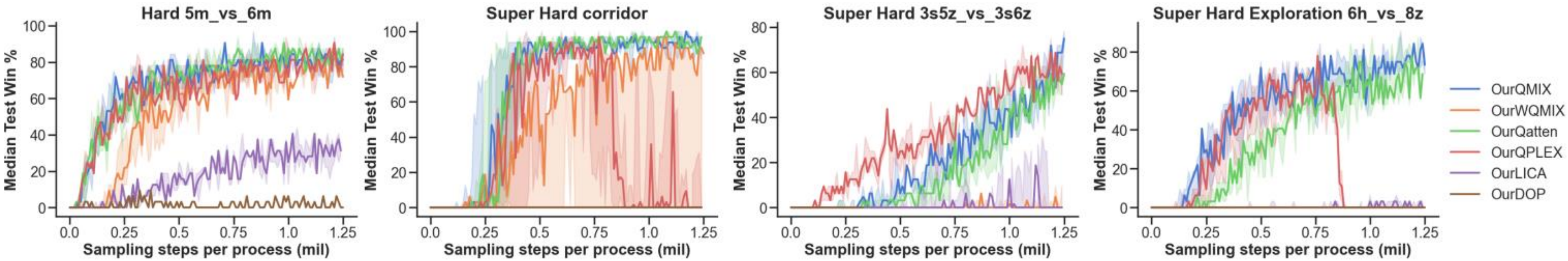


Figure 1: Median test win rate of SOTA MADRL algorithms.

Scenarios	*5m_vs_6m*	*3s5z_vs_3s6z*	*corridor*	*6h_vs_8z*
OurQMIX	**90%**	**75%**	**100%**	**84%**
OurQatten	**90%**	62%	**100%**	68%
OurQPLEX	**90%**	68%	96%	78%
OurWQMIX	**90%**	6%	96%	78%
OurLICA	40%	18%	0%	3%
OurDOP	9%	0%	0%	0%

Hu, Jian, et al. "Hyperparameter Tricks in Multi-Agent Reinforcement Learning: An Empirical Study." *arXiv e-prints* (2021): arXiv-2102.

Multi-Agent RL

Hyperparameter Tricks

- Result

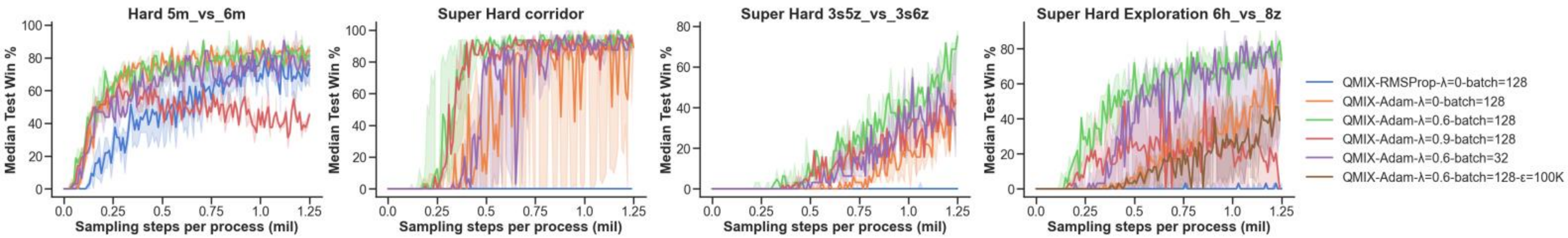


Figure 3: Ablation analysis of hyperparameters of QMIX.

Optimizer	Batch Size	λ	ε Anneal Steps	*5m_vs_6m*	*3s5z_vs_3s6z*	*corridor*	*6h_vs_8z*
RMSProp	128	0		84%	0%	0%	3%
Adam	128	0		**90%**	43%	98%	68%
Adam	128	0.6	500K for *6h_vs_8z*, 100K for others	**90%**	**75%**	**100%**	**84%**
Adam	32	0.6		**90%**	54%	96%	**84%**
Adam	128	0.9		64%	48%	98%	59%
Adam	128	0.6	100K	-	-	-	46%

Table 5: Ablation analysis of hyperparameters of QMIX. Adam, large batch size and Q(λ) improve the performance of QMIX significantly.

Hu, Jian, et al. "Hyperparameter Tricks in Multi-Agent Reinforcement Learning: An Empirical Study." *arXiv e-prints* (2021): arXiv-2102.