# 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

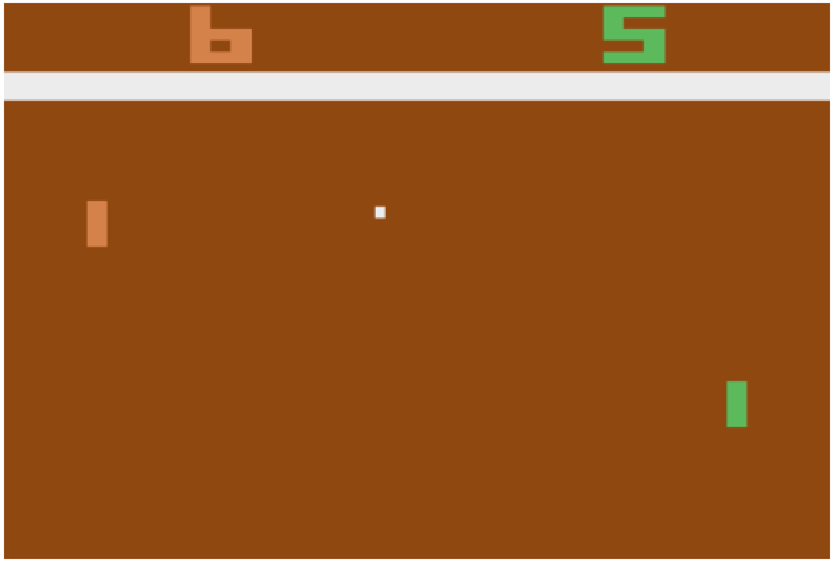
# Multi-Agent RL

- 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 $\rho$ to change gradually from -1 to +1

| | Left player scores | Right player scores |
|---|---|---|
| Left player reward | $\rho$ | $-1$ |
| Right player reward | $-1$ | $\rho$ |

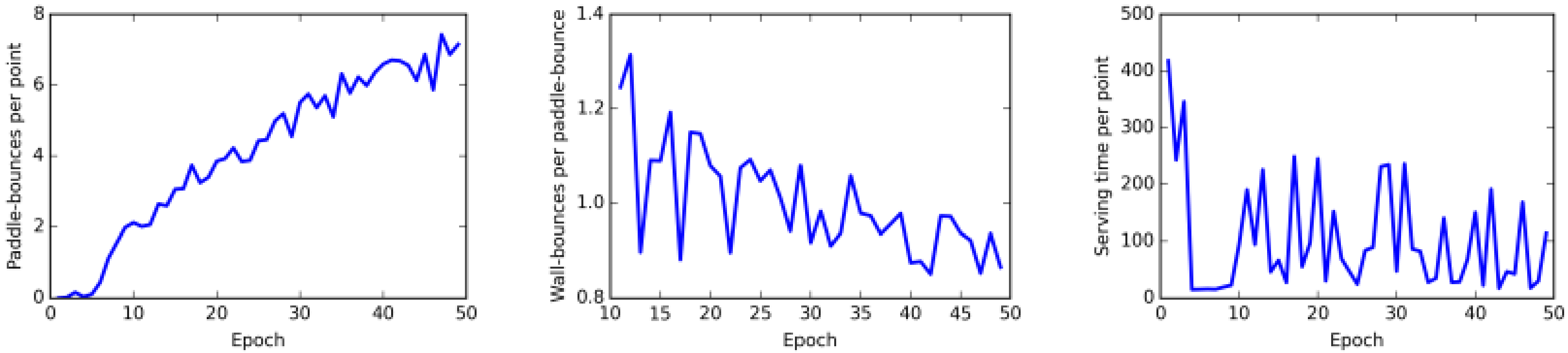
- Three measures
  - Average paddle-bounces per point
  - Average wall-bounces per paddle-bounce
  - Average serving time per point

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

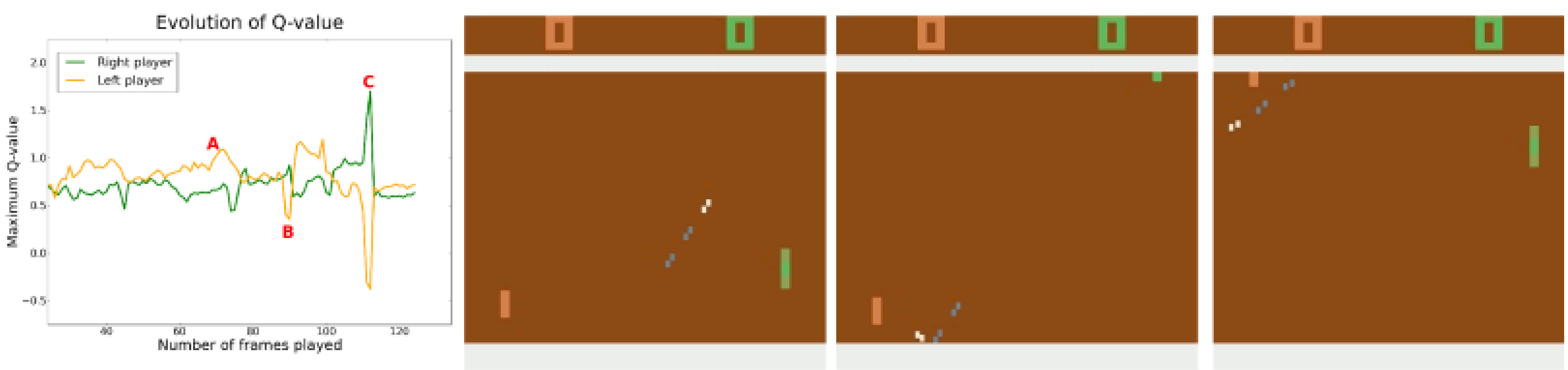
# Multi-Agent RL

## IQL

- Result—Fully Competitive




(a) Paddle-bounces per point (b) Wall-bounces per paddle-bounce (c) Serving time per point

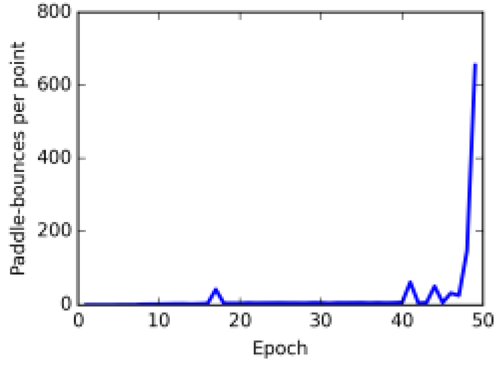



Tampuu, Ardi, et al. "Multiagent cooperation and competition with deep reinforcement learning." *PloS one* 12.4 (2017): e0172395.

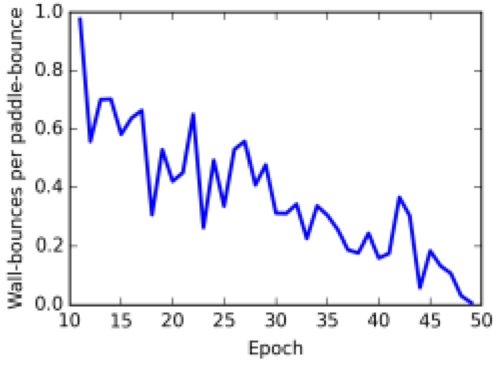
# Multi-Agent RL

## IQL

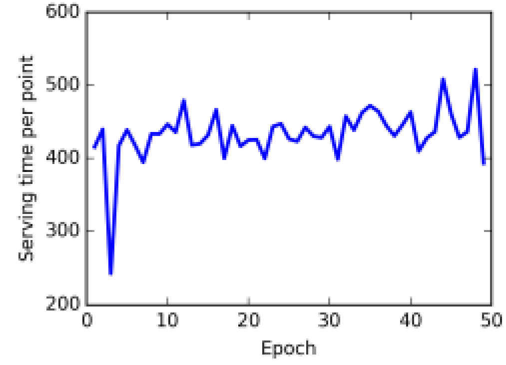
- Result—Fully Cooperative



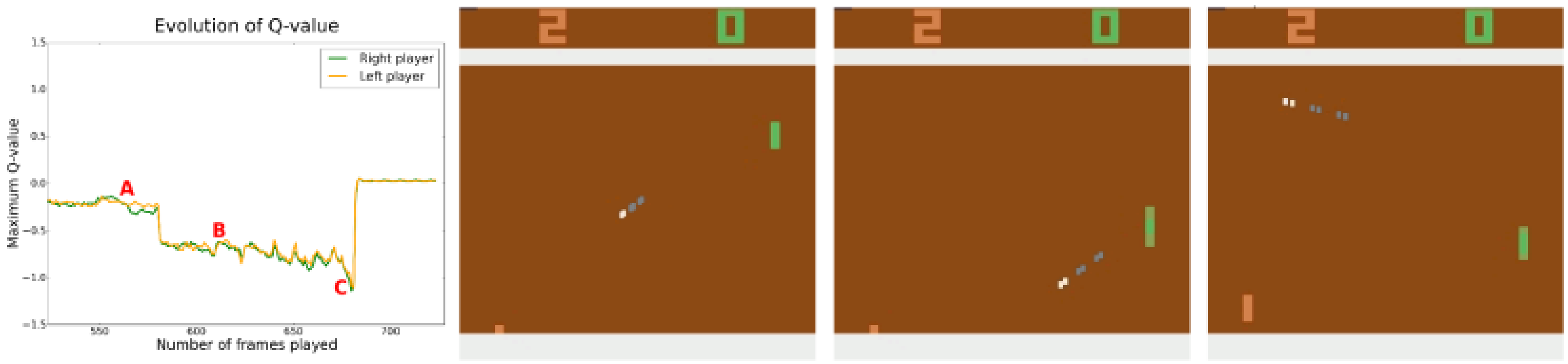

(a) Paddle-bounces per point




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

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

# Multi-Agent RL

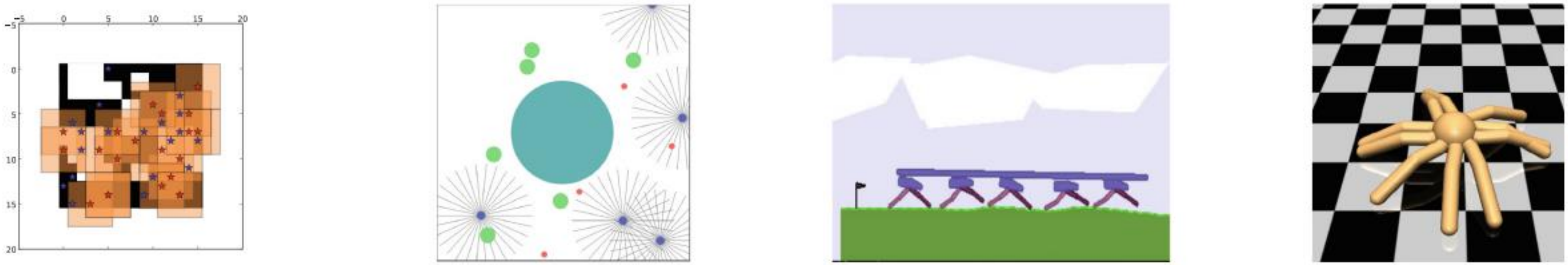
## IL with parameter sharing

- 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

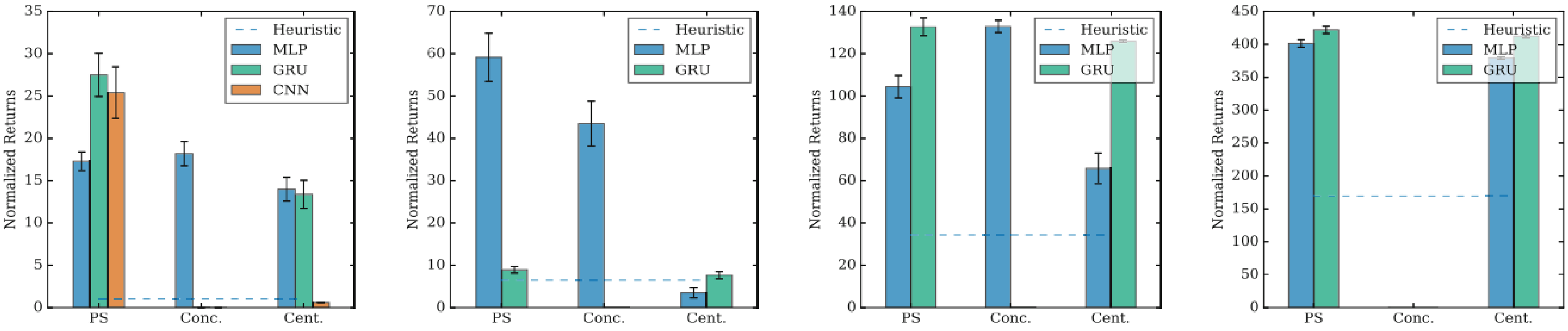
Gupta, Jayesh K., Maxim Egorov, and Mykel Kochenderfer. "Cooperative multi-agent control using deep reinforcement learning." *International conference on autonomous agents and multiagent systems*. Springer, Cham, 2017.

# 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 | $10.1 \pm 6.3$ | $25.5 \pm 5.4$ | $17.4 \pm 4.9$ |
| Waterworld | NA | $10.1 \pm 5.7$ | $49.1 \pm 5.7$ |
| Multiwalker | $-8.3 \pm 3.2$ | $12.4 \pm 6.1$ | $58.0 \pm 4.2$ |
| Multi-ant | $307.2 \pm 13.8$ | $483.4 \pm 3.4$ | $488.1 \pm 1.3$ |

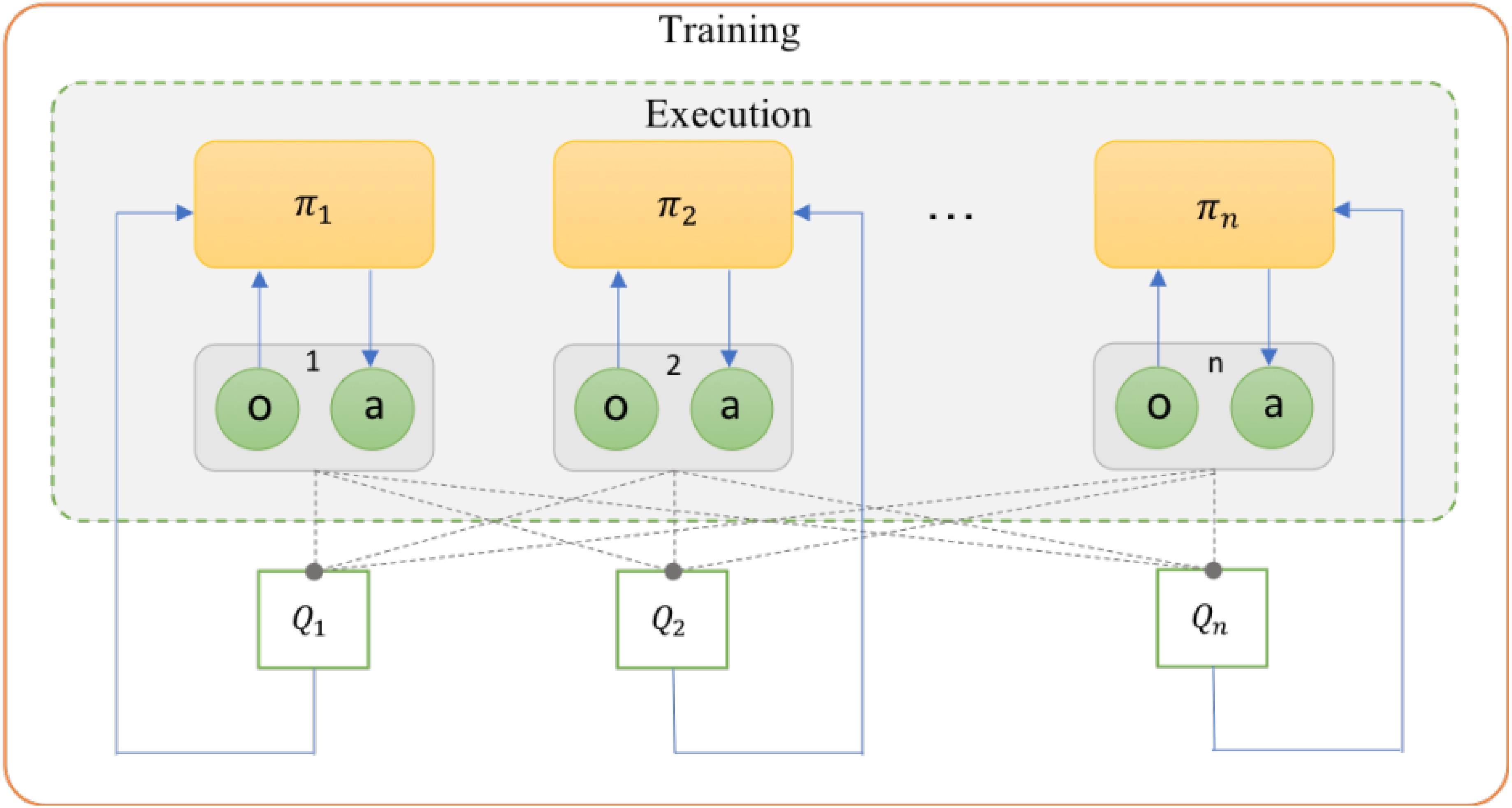
Gupta, Jayesh K., Maxim Egorov, and Mykel Kochenderfer. "Cooperative multi-agent control using deep reinforcement learning." *International conference on autonomous agents and multiagent systems*. Springer, Cham, 2017.

# 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

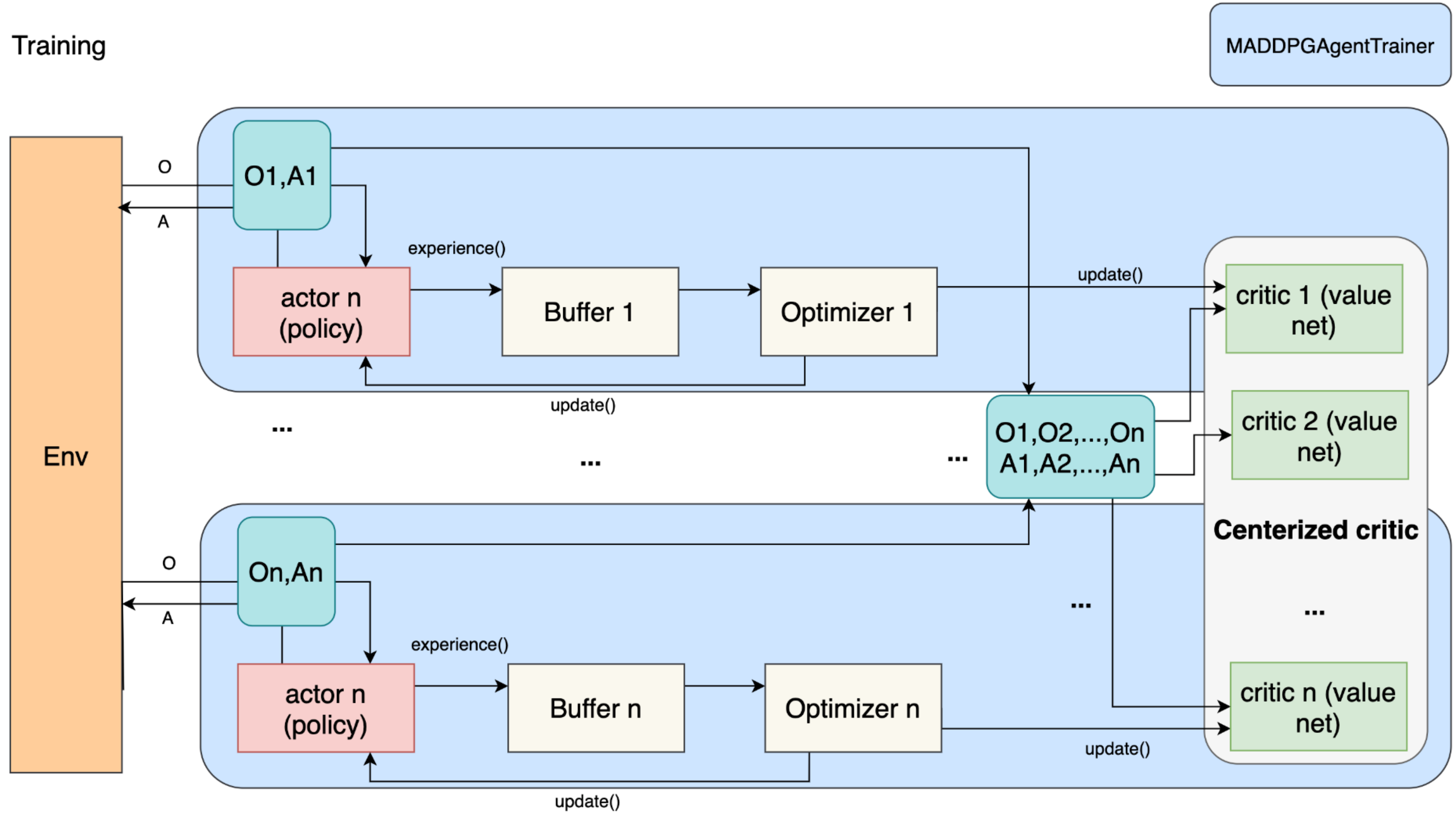
- Challenge
  - Non-stationarity of the environment
  - Policy gradient suffers from a variance that increases as the number of agents grows
- Main idea
  - Leads to learned policies that only use local information at execution time and allow the policies to use extra information to ease training
  - 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:

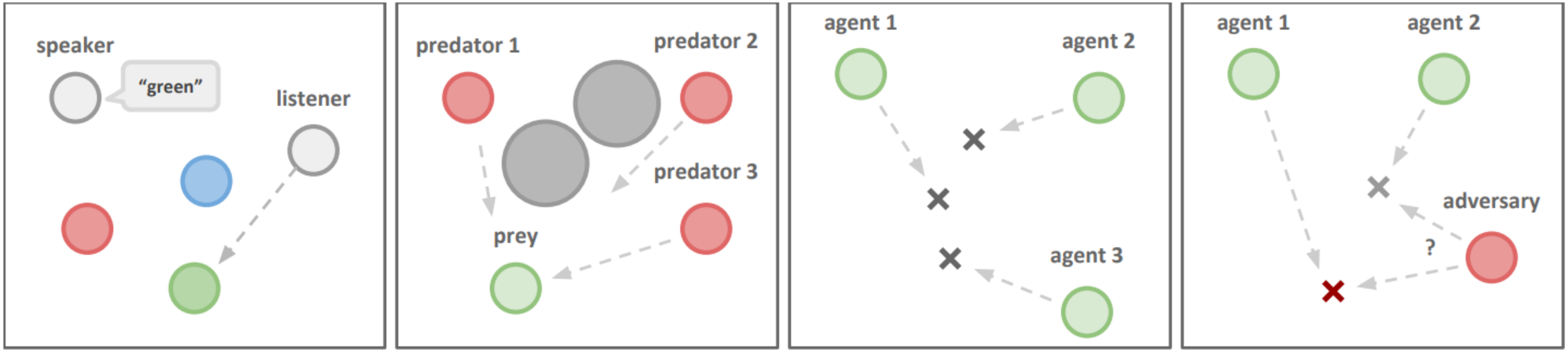
$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x}, a\sim\mathcal{D}}\left[\nabla_{\theta_i} \boldsymbol{\mu}_i(a_i \mid o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N)\Big|_{a_i = \boldsymbol{\mu}_i(o_i)}\right]$$

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

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

## MADDPG

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

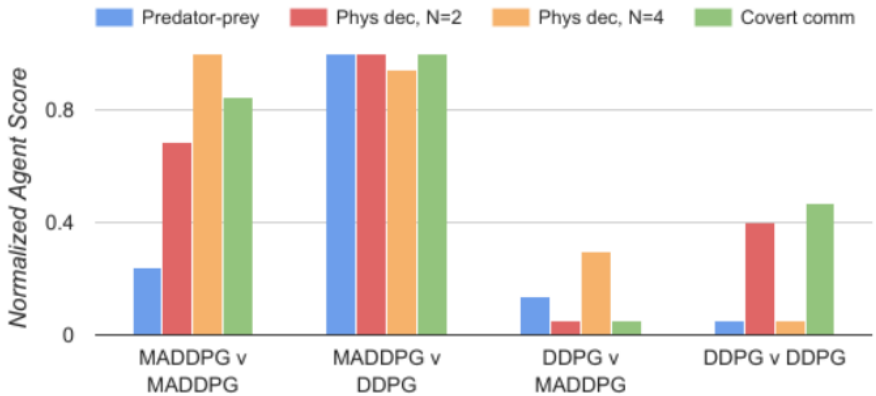



Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." *Advances in neural information processing systems* 30 (2017).

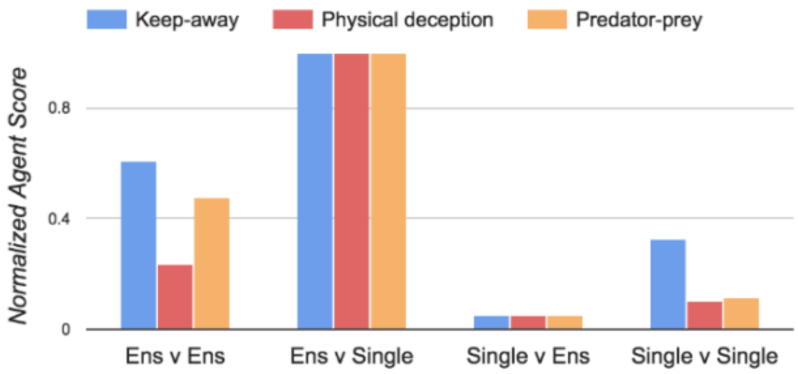
# Multi-Agent RL

## MADDPG

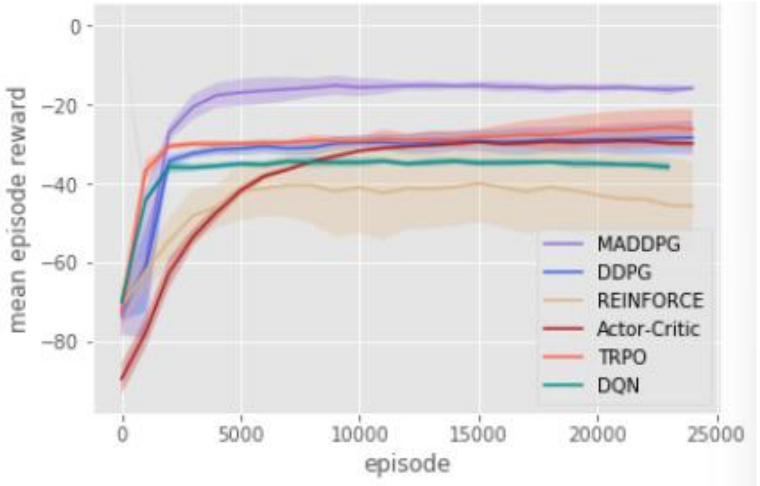
- Performance




Comparison of MADDPG and DDPG (Agent v Adversary)




Comparison of Ensemble and Single Policies (Agent v Adversary)





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.

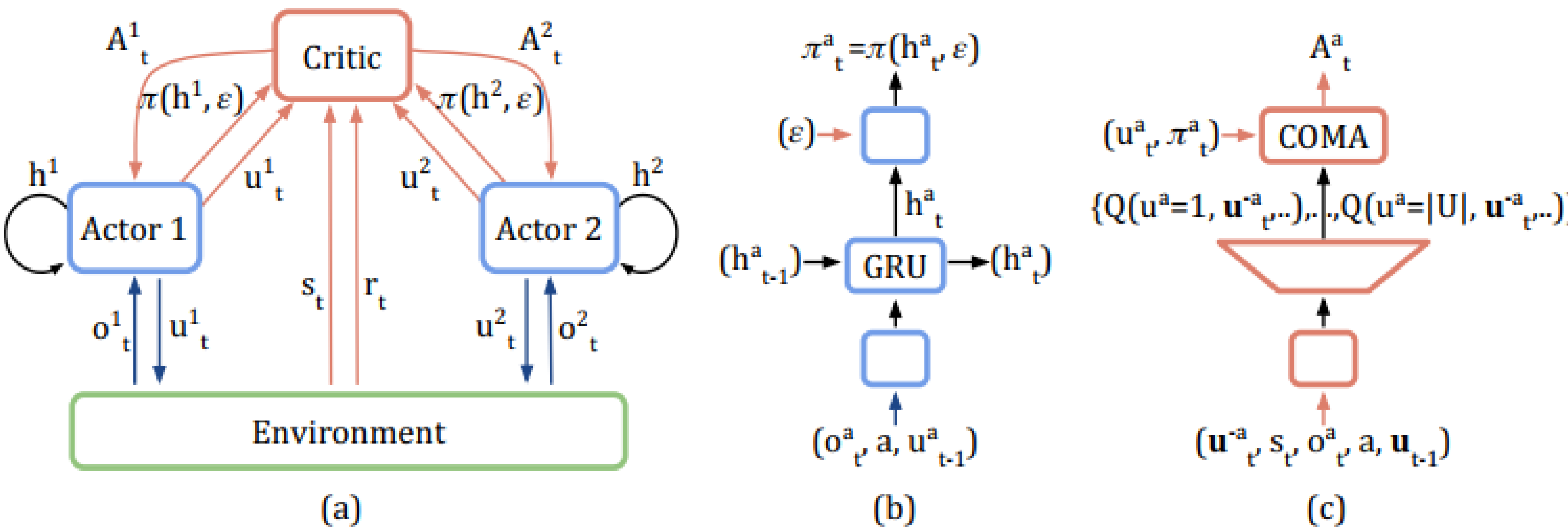
# Multi-Agent RL

## 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

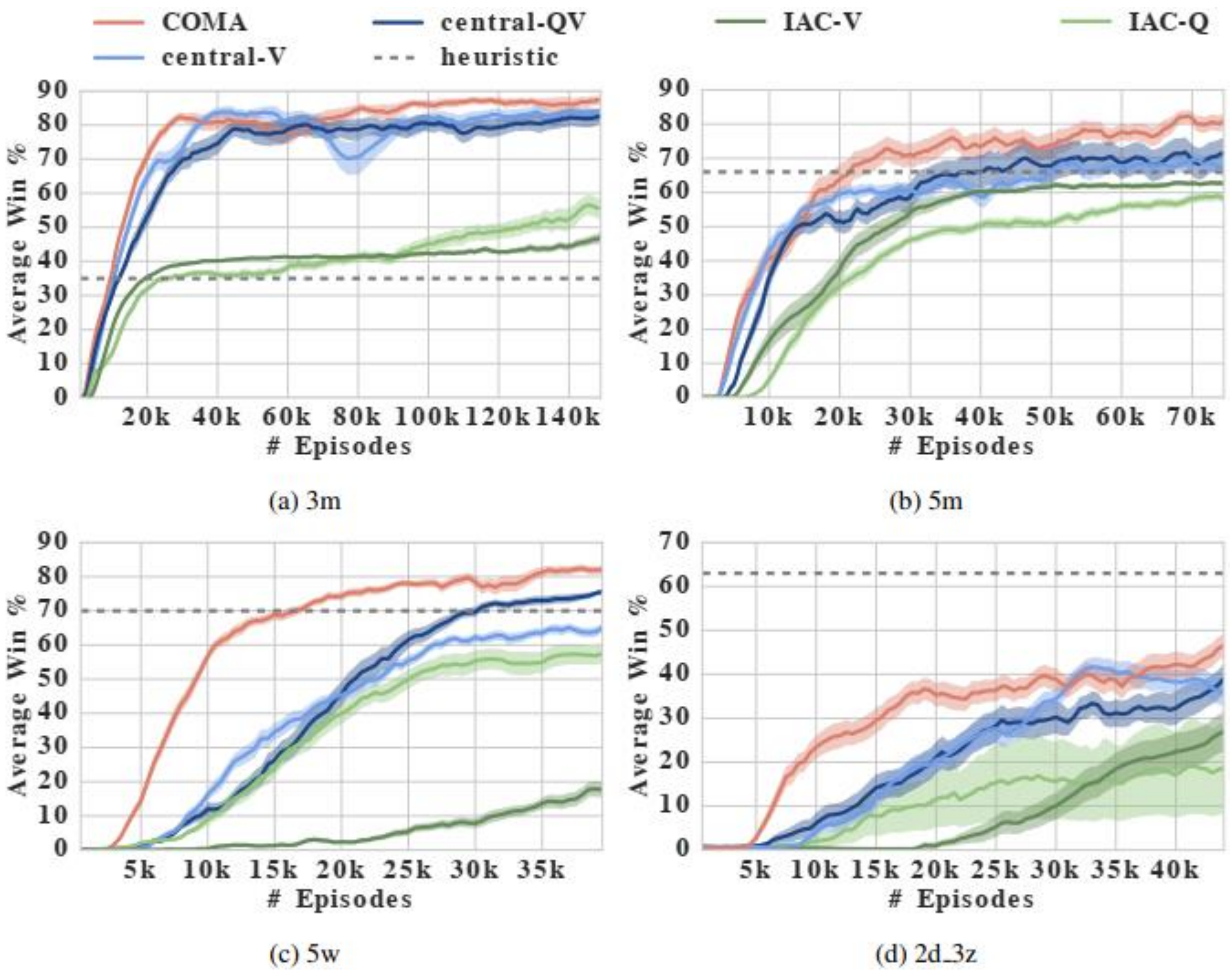
**Algorithm 1** Counterfactual Multi-Agent (COMA) Policy Gradients

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($\lambda$) 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, $r(\mathbf{s},\mathbf{a}) = r_1(o_t^1,a_t^1) + r_2(o_t^2,a_t^2)$

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

- 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

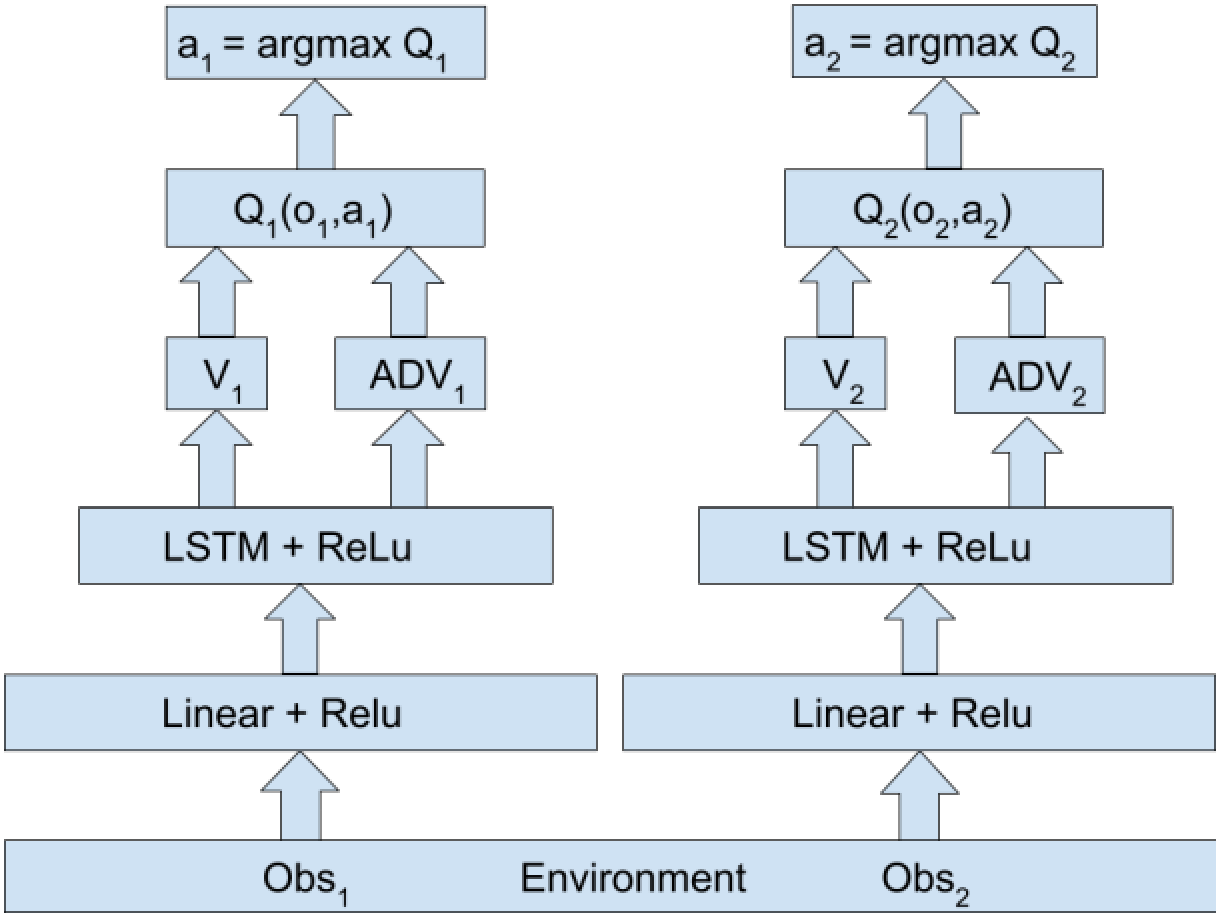



Figure 14: Independent Agents Architecture

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

# Multi-Agent RL

- VDN
  - Architecture

VDN

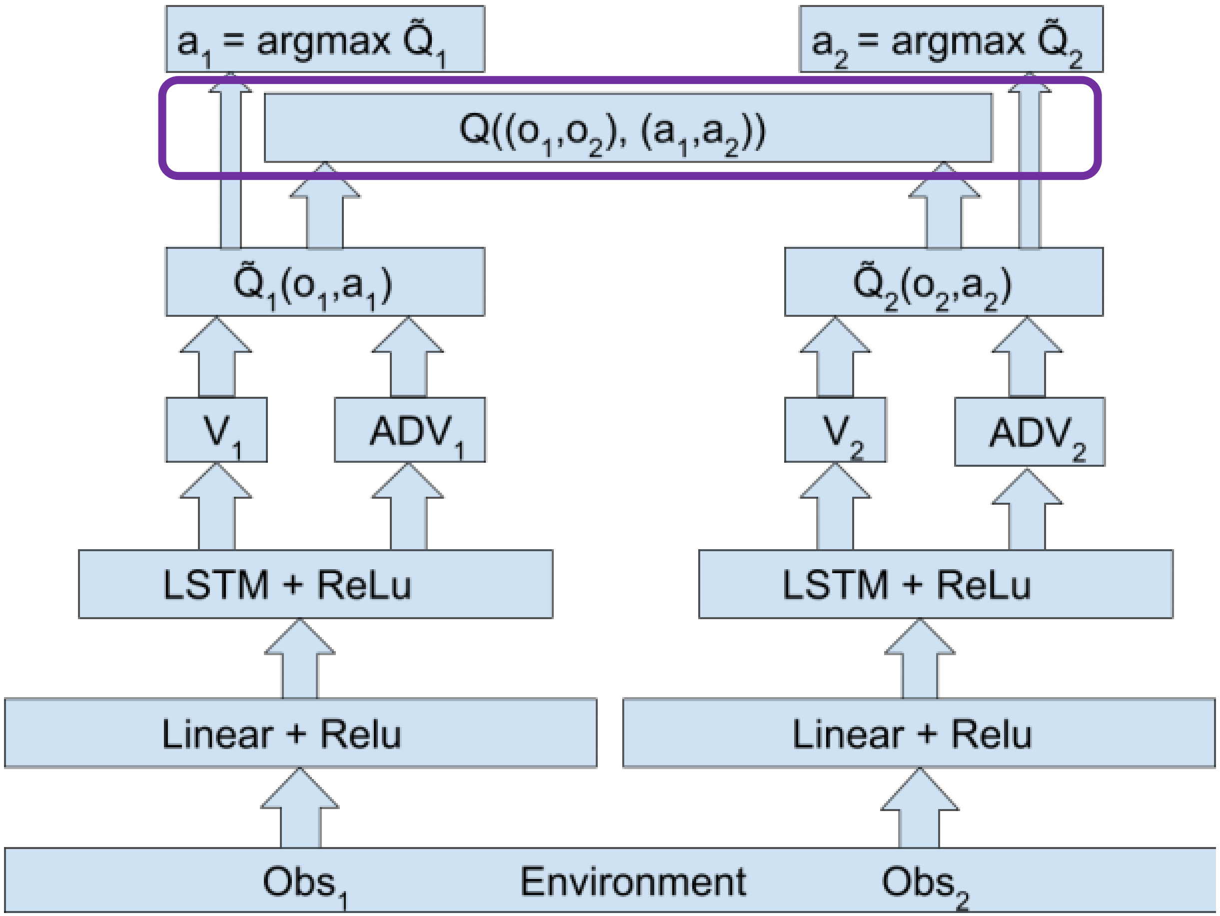



Figure 15: Value-Decomposition Individual Architecture

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

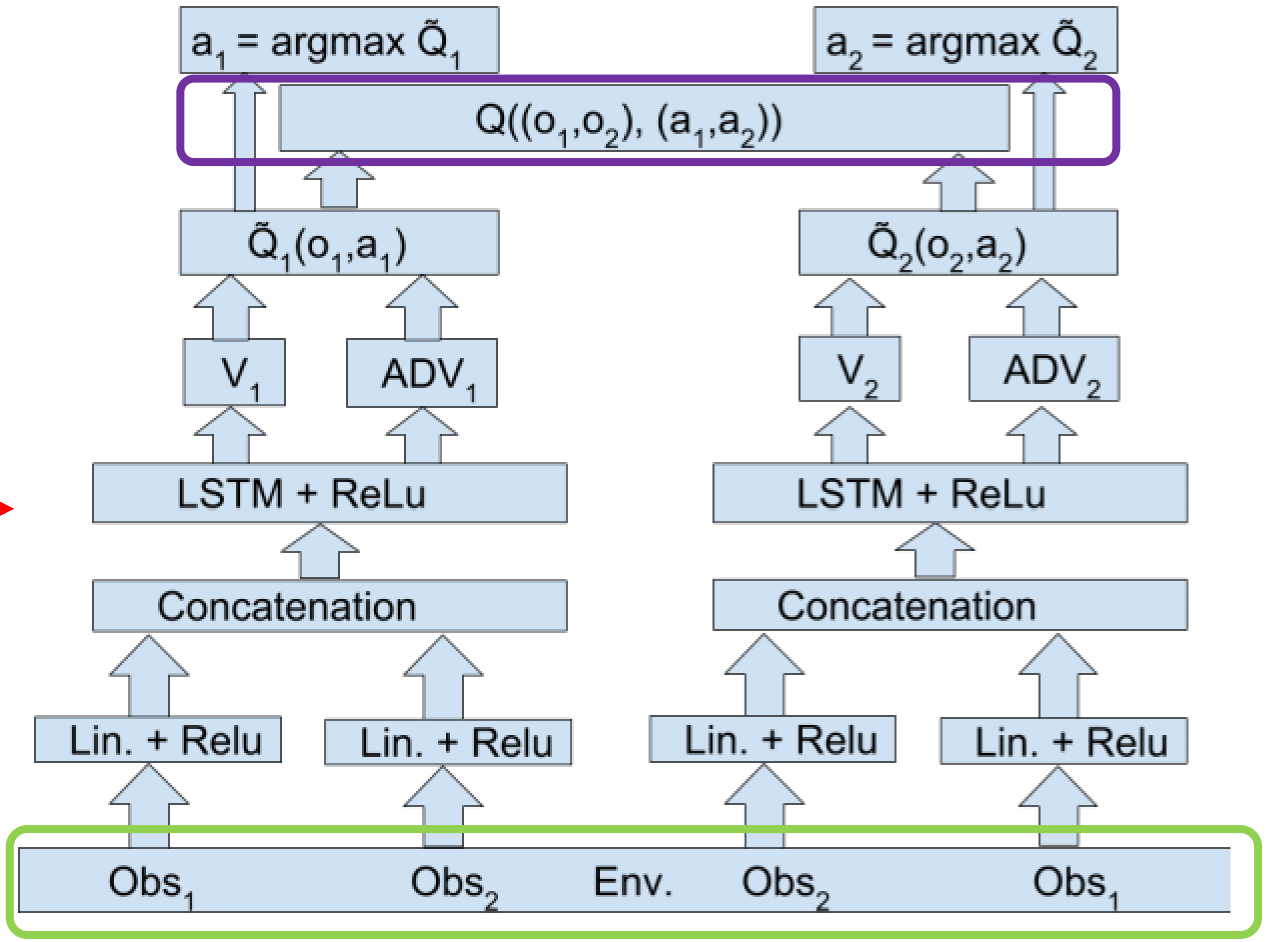



Figure 16: Low-level communication Architecture

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

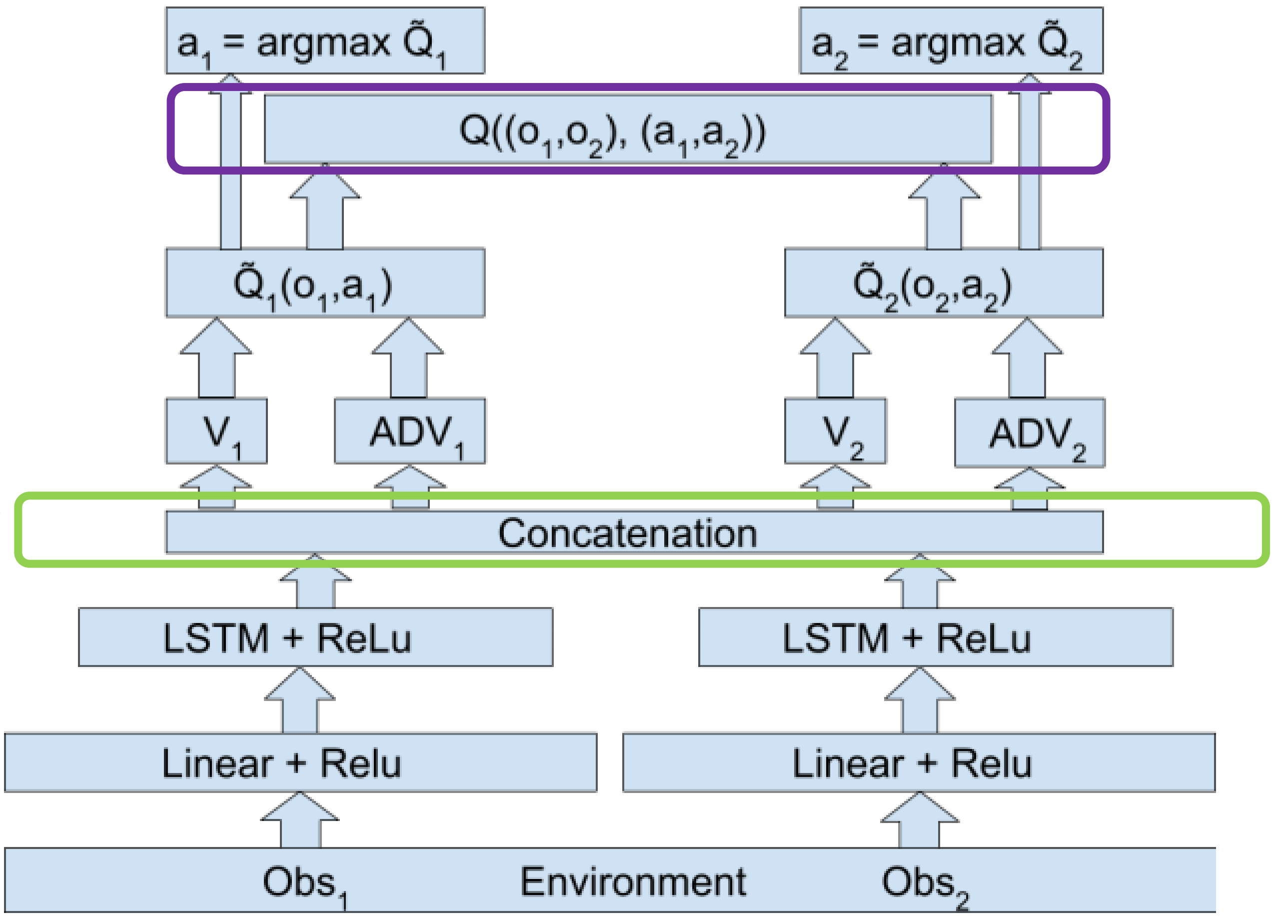



Figure 17: High-level communication Architecture

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

# Multi-Agent RL

- **VDN**
  - Architecture

Centralized

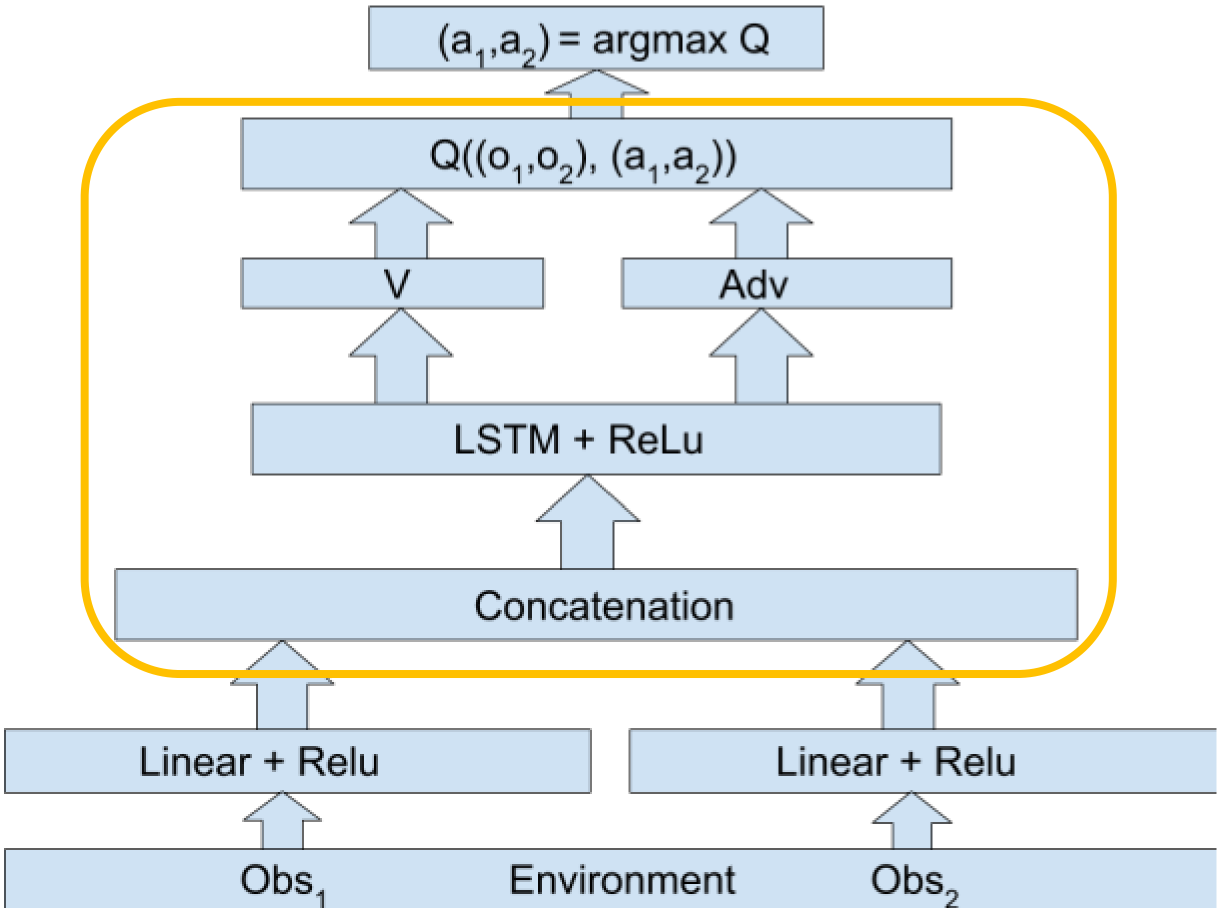



Figure 20: Combinatorially Centralized Architecture

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

## VDN

- Performance

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

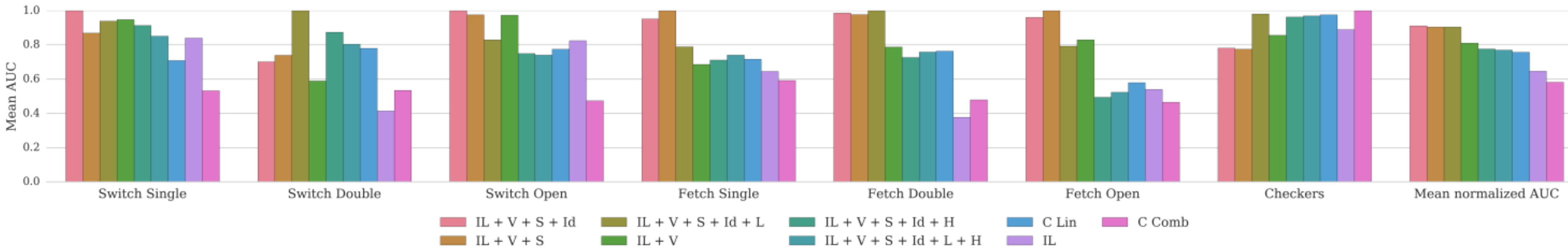
Table 1: Agent architectures. V is value decomposition, S means shared weights and an invariant network, Id means role info was provided, L stands for lower-level communication, H for higher-level communication and C for centralization. These architectures were selected to show the advantages of the 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

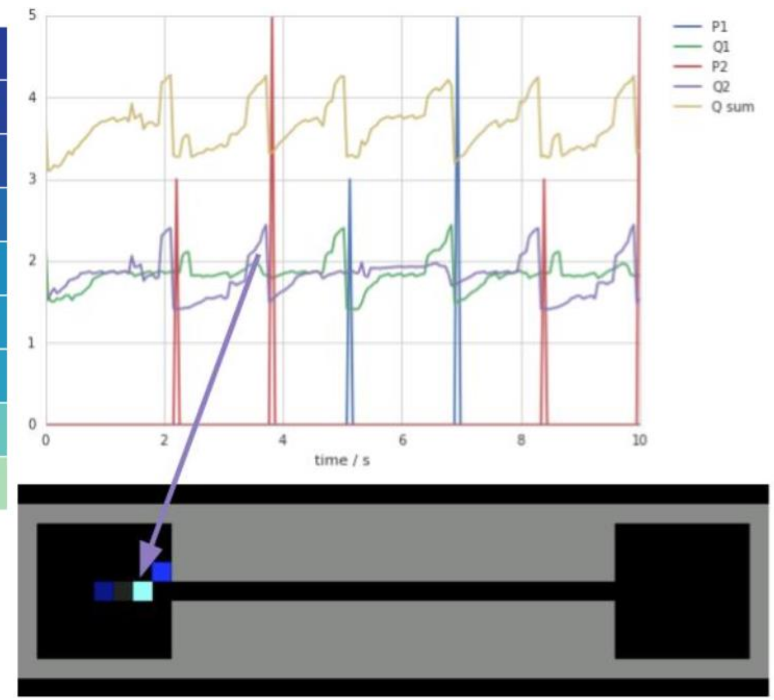
## VDN

### ➢ Performance




| Architecture | Checkers | Fetch Double | Fetch Open | Fetch Single | Switch Double | Switch Open | Switch Single | Mean final reward |
|---|---|---|---|---|---|---|---|---|
| IL + V + S + Id + L | 0.96 | 0.98 | 0.79 | 0.82 | 1.00 | 0.84 | 0.97 | 0.91 |
| IL + V + S + Id | 0.78 | 1.00 | 0.85 | 0.96 | 0.69 | 1.00 | 1.00 | 0.90 |
| IL + V + S | 0.75 | 0.90 | 1.00 | 1.00 | 0.63 | 0.96 | 0.91 | 0.88 |
| IL + V | 0.88 | 0.86 | 0.76 | 0.70 | 0.66 | 0.94 | 0.98 | 0.83 |
| IL + V + S + Id + L + H | 0.95 | 0.82 | 0.49 | 0.78 | 0.80 | 0.72 | 0.85 | 0.77 |
| IL + V + S + Id + H | 0.95 | 0.70 | 0.45 | 0.72 | 0.86 | 0.68 | 0.93 | 0.76 |
| C Lin | 0.96 | 0.86 | 0.47 | 0.73 | 0.68 | 0.76 | 0.72 | 0.74 |
| IL | 0.92 | 0.38 | 0.47 | 0.66 | 0.44 | 0.82 | 0.83 | 0.64 |
| C Comb | 1.00 | 0.46 | 0.38 | 0.53 | 0.60 | 0.45 | 0.54 | 0.57 |

Task




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 $Q_a$

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

  - 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.

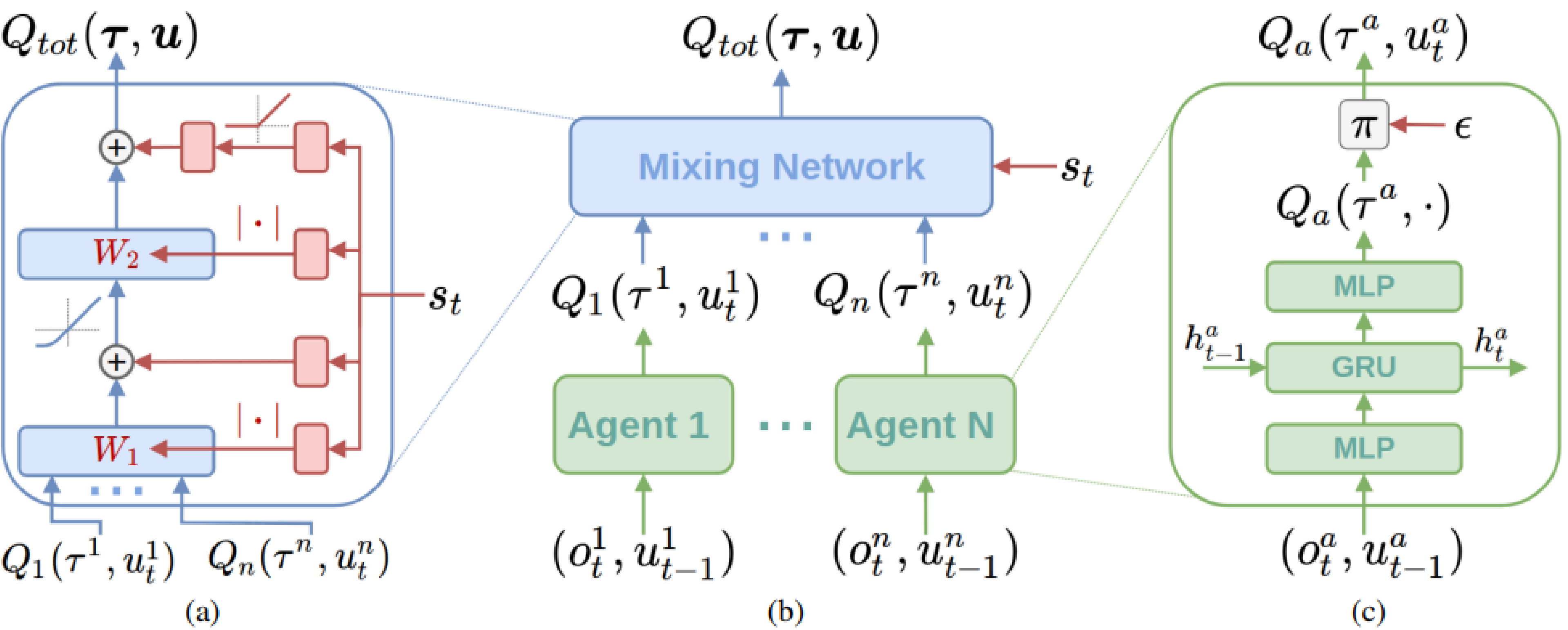
Rashid, Tabish, et al. "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning." *International Conference on Machine Learning*. PMLR, 2018.

# Multi-Agent RL

## QMIX

- Architecture
- Loss

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

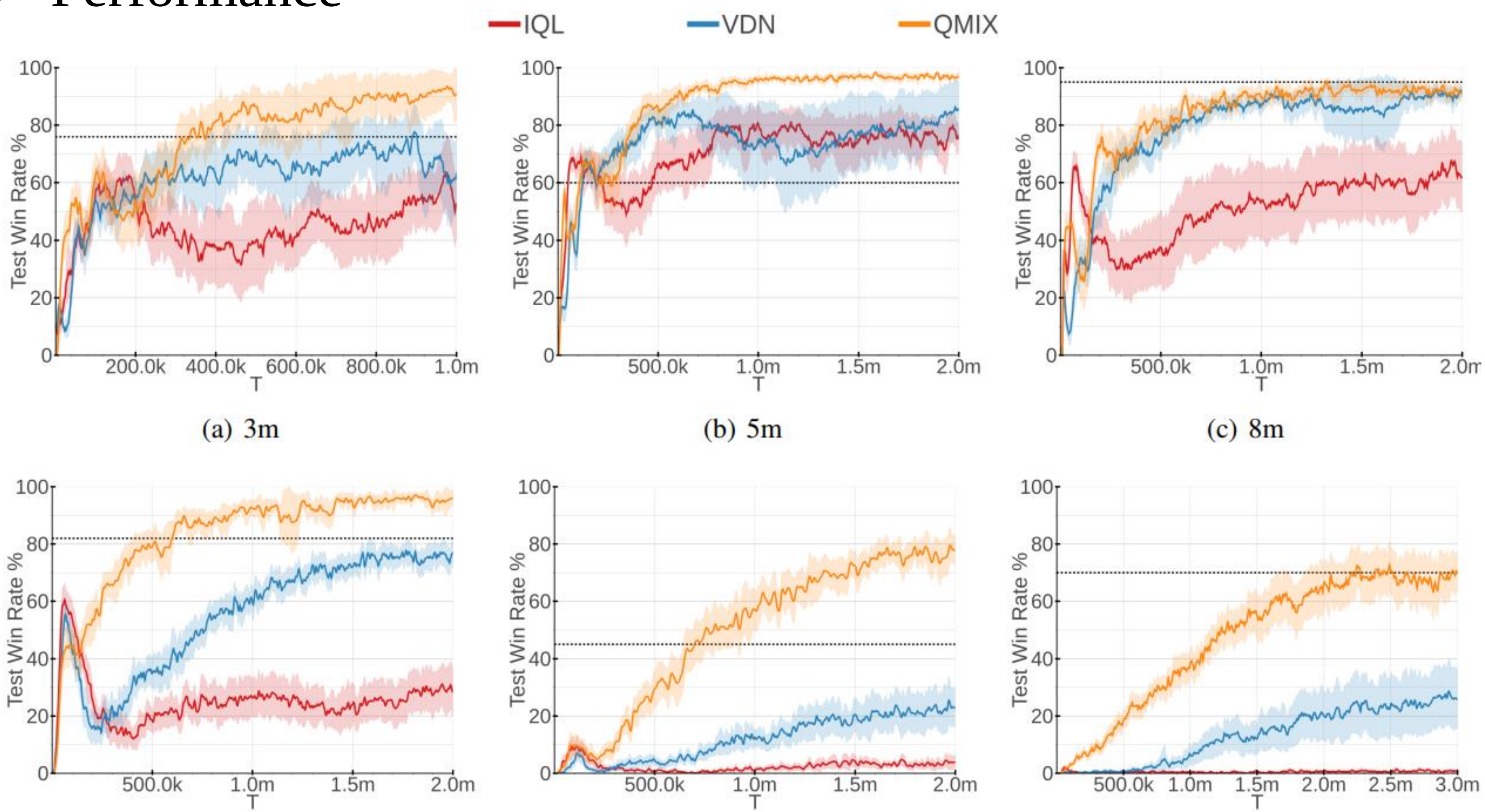



Rashid, Tabish, et al. "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning." *International Conference on Machine Learning*. PMLR, 2018.

# Multi-Agent RL

## QMIX

- Performance




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

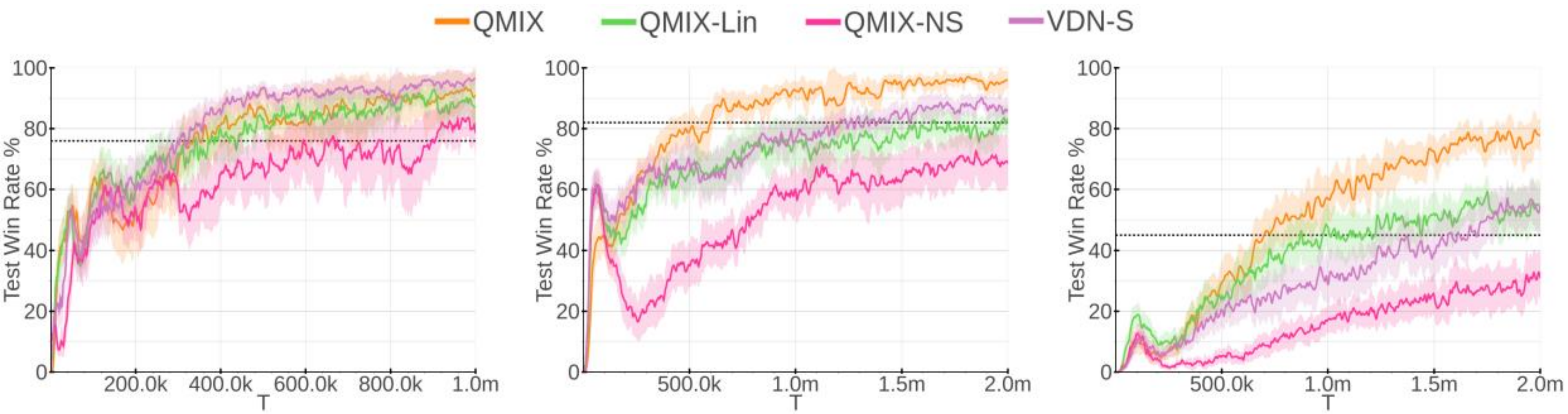
(d) 2s_3z (e) 3s_5z (f) 1c_3s_5z

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.