NeurIPS 2021

Co-Adaptation of Algorithmic and Implementational Innovations in Inference-based Deep Reinforcement Learning

Hiroki Furuta[1], Tadashi Kozuno[2], Tatsuya Matsushima[1],

Yutaka Matsuo[1], Shixiang Shane Gu[3]

[1]The University of Tokyo, [2]University of Alberta, [3]Google Brain

Contact: furuta@weblab.t.u-tokyo.ac.jp

Popular Inference-based Methods

Interpreting RL as probabilistic inference yields many kinds of algorithms

E.g.

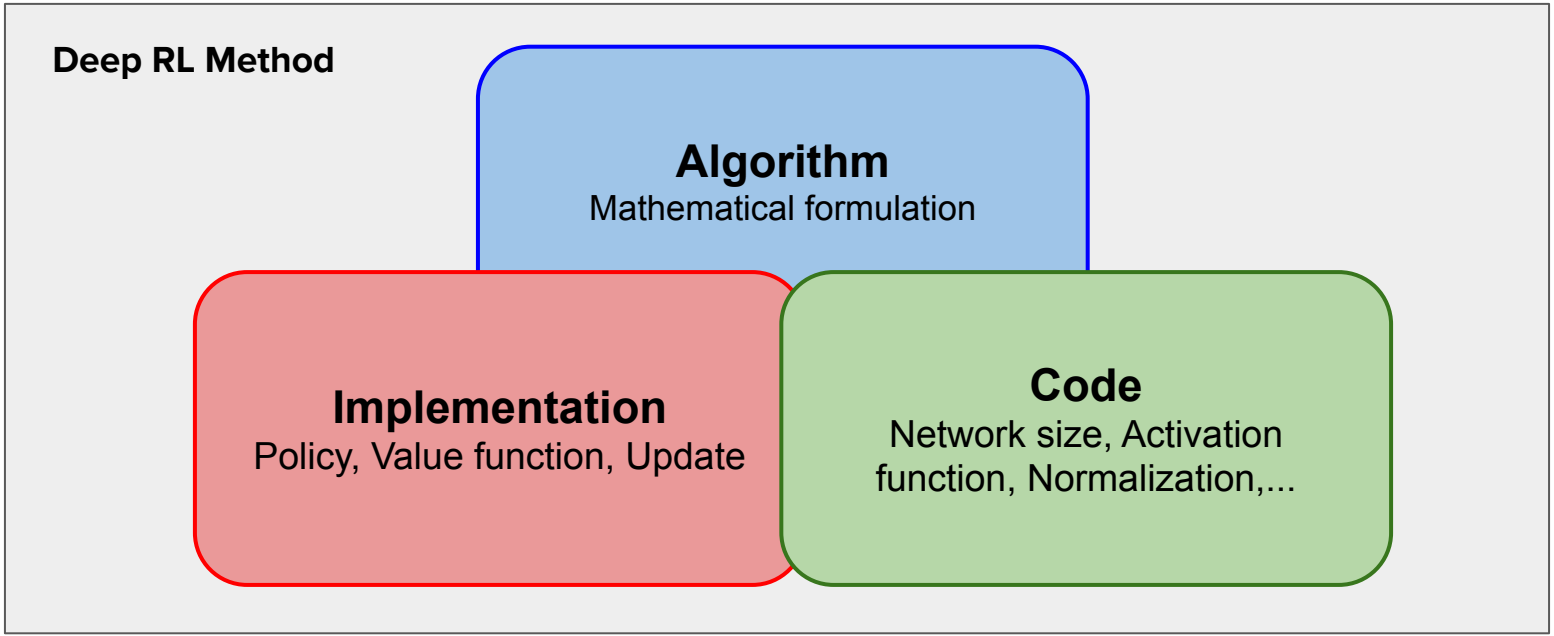
- **MPO** [Abdolmaleki et al. 2018]: “Pseudo-likelihood” objective (inspired by REPS)
- **AWR** [Peng et al. 2019]: “Pseudo-likelihood” objective and stable in offline RL
- **SAC** [Haarnoja et al. 2018]: Maximum entropy objective and soft Q function

They have similar deviations to each other, but **low-level features** (implementation design choice, code details) and **high-level design** (algorithm) are very different

Co-Adaptation of Deep RL Algorithms

Mixture of these components makes it difficult to identify the performance gaps

We successfully distinguish between co-adapted and no-co-adapted design choice



Meta Analyses of RL Algorithms

Prior Contributions

Henderson et al. 2017: Point out high-variance results across implementation/seeds

Tucker et al. 2018: High performances of action-dependent baselines depends on subtle implementation choices

Engstrom et al. 2019: Compare PPO/TRPO and optimize code-level designs

Andrychowicz et al. 2021: List up low & high level design choices in on-policy algorithms, and combine them to PPO in a large-scale evaluations

Ours: Focus on two distinct families of **off-policy algorithms**, provide **mathematical connections**, evaluate **co-adaptive implementation** and **code** design choices

Inference-based Methods & Unified Objective

As in previous works [Levine 2018, Abdolmaleki et al. 2018], we can consider the marginal log-likelihood of optimality variables and its decomposition

$$\log \Pr\left(\mathcal{O}=1|\pi_p\right) = \mathbb{E}_q\left[\log \Pr(\mathcal{O}=1|\tau) - \log\frac{q(\tau)}{p(\tau)} + \log\frac{q(\tau)}{p(\tau|\mathcal{O}=1)}\right]$$
$$= \mathcal{J}(p,q) + D_{KL}(q(\tau)\,||\,p(\tau|\mathcal{O}=1)),$$

Inference-based methods aim to find the parametric policy that maximize ELBO

ELBO: $$\mathcal{J}(p,q) := \mathbb{E}_q\left[\log \Pr(\mathcal{O}=1|\tau)\right] - D_{KL}(q(\tau)\,||\,p(\tau))$$

$$p(\tau) = p(s_1)\prod_t p(s_{t+1}|s_t,a_t)\pi_p(a_t|s_t), \qquad q(\tau) = p(s_1)\prod_t p(s_{t+1}|s_t,a_t)\pi_q(a_t|s_t),$$

Inference-based Methods & Unified Objective

Probability of optimality variable

$$\Pr(\mathcal{O} = 1|\tau) \propto \exp\left(\sum_{t=1}^{T} \eta^{-1}\mathcal{G}(s_t, a_t)\right)$$

G is a function over state × action space (immediate reward, Q-function, advantage)

With some relaxations (one-step, infinite-horizon, etc...), we can derive **unified objective** for recent popular SoTA off-policy algorithms (SAC, MPO, AWR)

$$\mathcal{J}(\pi_p, \pi_q) = \mathbb{E}_{d_\pi(s)}[\eta^{-1}\mathcal{G}(s, a) - D_{KL}(\pi_q \,||\, \pi_p)]$$

$$\max_{\pi_p, \pi_q} \mathcal{J}(\pi_p, \pi_q) \;\text{ s.t. } \int d_\pi(s) \int \pi_p(a|s)\, dads = 1 \text{ and } \int d_\pi(s) \int \pi_q(a|s)\, dads = 1$$

Expectation-Maximization (EM) Control (Algorithm)

Solve unified objective via Expectation-Maximization algorithms

E-step: Get non-parametric policy $\mathcal{J}(\pi_{\theta_p^{(k-1)}}, \pi_q)$

$$\mathcal{J}(\pi_q, \beta) = \int d_\pi(s) \int \pi_q(a|s) \frac{\mathcal{G}(s,a)}{\eta} \, dads$$
$$- \int d_\pi(s) \int \pi_q(a|s) \log \frac{\pi_q(a|s)}{\pi_{\theta_p^{(k-1)}}(a|s)} \, dads + \beta \left(1 - \int d_\pi(s) \int \pi_q(a|s) \, dads\right)$$

$$\pi_q^{(k)}(a|s) = Z(s)^{-1} \pi_{\theta_p^{(k-1)}}(a|s) \exp\left(\eta^{-1} \mathcal{G}(s,a)\right)$$

M-step: Project E-step policy to the parametric policy $\mathcal{J}(\pi_q^{(k)}, \pi_p)$

$$\max_{\theta_p} \mathbb{E}_{d_\pi(s)\pi_q^{(k)}(a|s)} \left[\log \pi_{\theta_p}(a|s)\right] = \max_{\theta_p} \mathbb{E}_{d_\pi(s)\pi_{\theta_p^{(k-1)}}(a|s)} \left[\frac{\log \pi_{\theta_p}(a|s)}{Z(s)} \exp\left(\eta^{-1} \mathcal{G}(s,a)\right)\right]$$

Direct KL Divergence Minimization Control (Algorithm)

Fix prior policy and only optimize variational posterior policy

- KL-Control is a sub-problem of EM-Control (corresponding to E-step)
- KL-Control converges to regularized-optimal policy, while EM-Control converges to standard-optimal policy

$$\max_{\theta_q} \mathbb{E}_{d_\pi(s)\pi_{\theta_q}(a|s)} \left[\eta^{-1} \mathcal{G}(s,a) - \log \frac{\pi_{\theta_q}(a|s)}{\pi_p(a|s)} \right]$$

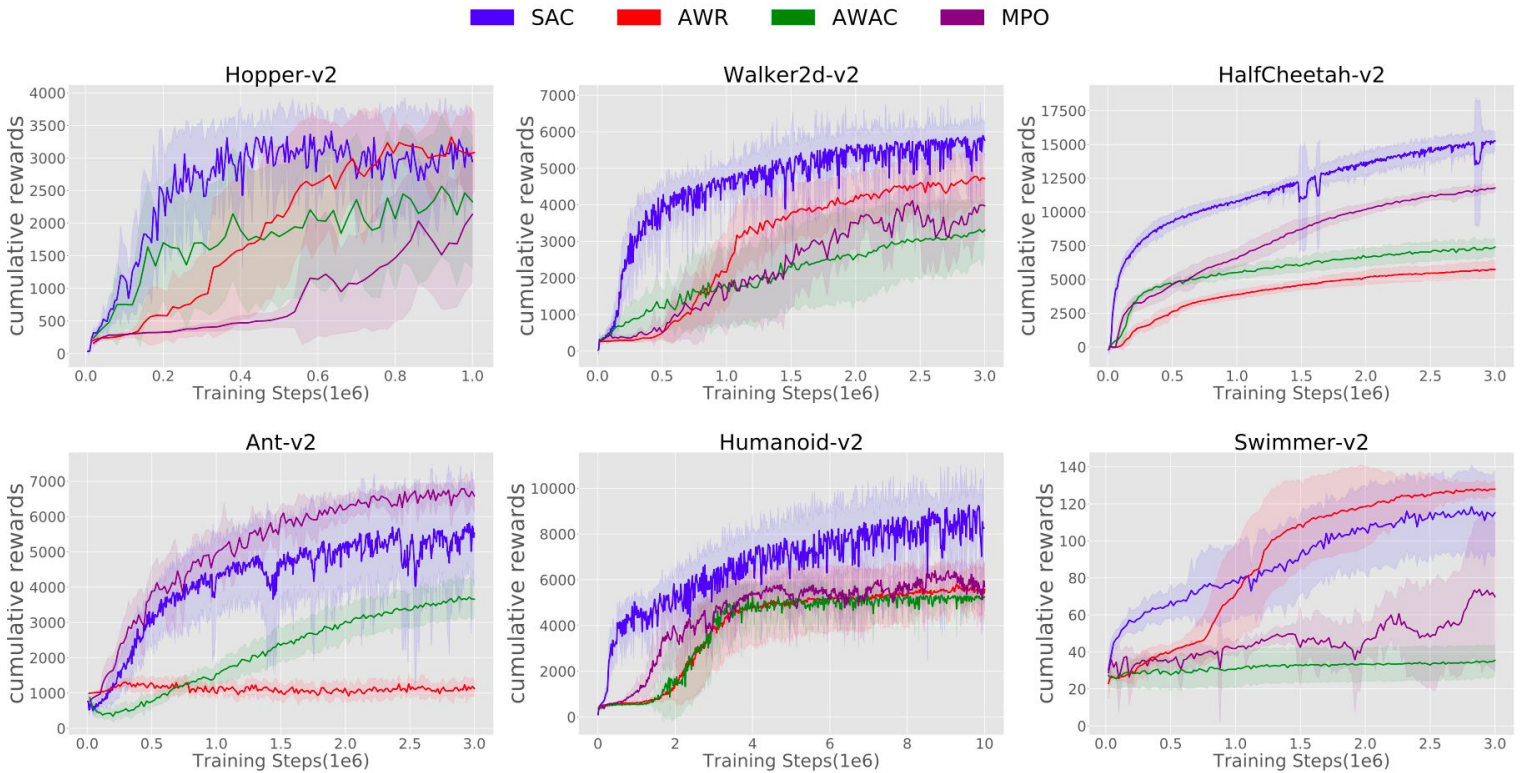
Taxonomy of Inference-based Algorithms

- **Algorithm**: EM-Control or KL-Control
- **Implementation**: Policy Updates, Value function & Estimation, Action distribution
- **Code**: Network size, Activation function, Normalization, etc ...

Method	Algorithm	Implementation				
		π_q update	π_p update	$\mathcal{G}$	$\mathcal{G}$ estimate	π_θ
MPO	EM	Analytic + TR	SG + TR	Q^{π_p}	Retrace(1)	$\pi_p = \mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$
AWR	EM	Analytic	Mixture + SG	$A^{\tilde{\boldsymbol{\pi}}_{\boldsymbol{p}}}$	TD(λ)	$\pi_p = \mathcal{N}(\mu_\theta(s), \Sigma)$
AWAC	EM	Analytic	Mixture + SG	Q^{π_p}	TD(0)	$\pi_p = \mathcal{N}(\mu_\theta(s), \Sigma_\theta)$
SAC	KL	SG	(Fixed to Unif.)	$Q^{\boldsymbol{\pi}_{\boldsymbol{q}}}_{\text{soft}}$	TD(0) + TD3	$\pi_q = \text{Tanh}(\mathcal{N}(\mu_\theta(s), \Sigma_\theta(s)))$
PoWER	EM	Analytic	Analytic	$\eta \log Q^{\pi_p}$	TD(1)	$\pi_p = \mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$
RWR	EM	Analytic	SG	$\eta \log r$	–	$\pi_p = \mathcal{N}(\mu_\theta(s), \Sigma)$
REPS	EM	Analytic	π_q	A^{π_p}	TD(0)	$\pi_p = \text{Softmax}$
UREX	EM	Analytic	SG	Q^{π_p}	TD(1)	$\pi_p = \text{Softmax}$
V-MPO	EM	Analytic + TR	SG + TR	A^{π_p}	n-step TD	$\pi_p = \mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$
TRPO	KL	TR	π_q	A^{π_p}	TD(1)	$\pi_q = \mathcal{N}(\mu_\theta(s), \Sigma_\theta)$
PPO	KL	SG + TR	π_q	A^{π_p}	GAE	$\pi_q = \mathcal{N}(\mu_\theta(s), \Sigma_\theta)$
DDPG*	KL	SG	(Fixed)	Q^{π_q}	TD(0)	$\pi_q = \mu_\theta(s)$
TD3*	KL	SG	(Fixed)	Q^{π_q}	TD(0) + TD3	$\pi_q = \mu_\theta(s)$

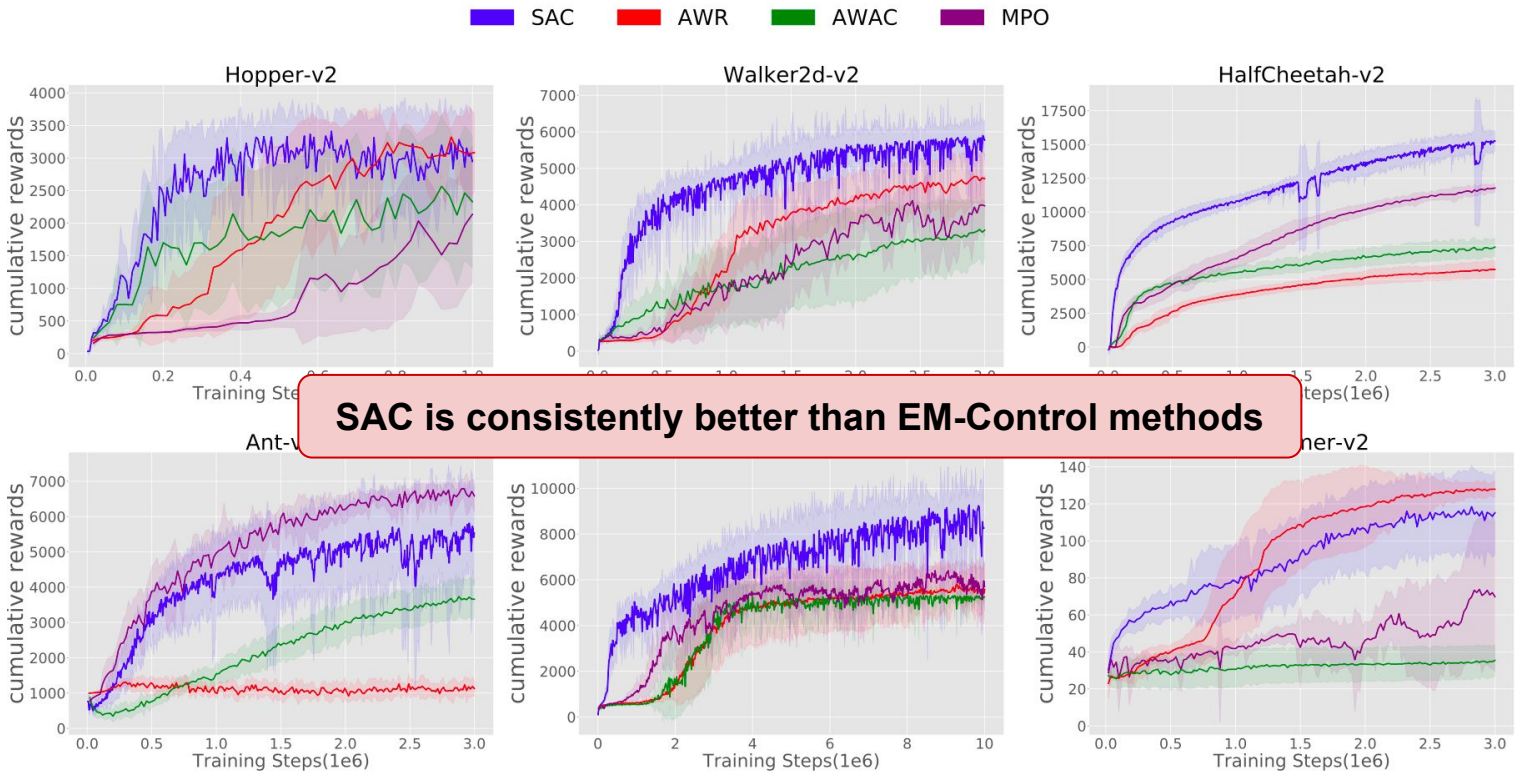
Empirical Comparisons

Open AI Gym MuJoCo 6 tasks (See appendix for DM Control 28 tasks)



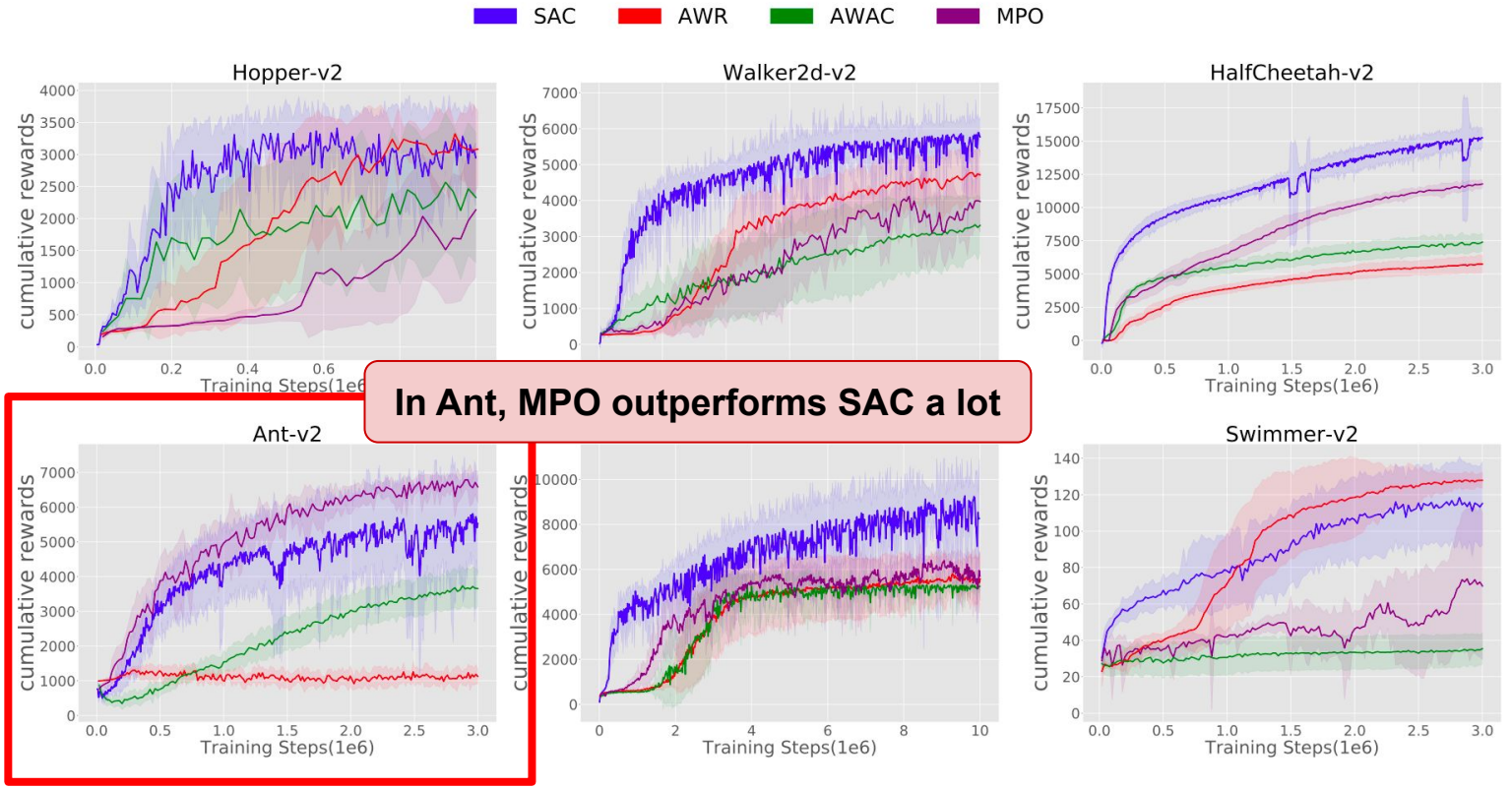
Empirical Comparisons

Open AI Gym MuJoCo 6 tasks (See appendix for DM Control 28 tasks)



Empirical Comparisons

Open AI Gym MuJoCo 6 tasks (See appendix for DM Control 28 tasks)



Clipped Double Q-Learning (Implementation)

(D): with clipped double Q-Learning [Fujimoto et al. 2018]

(S): without clipped double Q-Learning

	SAC (D)	SAC (S)	AWAC (D)	AWAC (S)	MPO (D)	MPO (S)
Hopper-v2	**3013 ± 602**	1601 ± 733	2329 ± 1020	2540 ± 755	2352 ± 959	2136 ± 1047
Walker2d-v2	**5820 ± 411**	1888 ± 922	3307 ± 780	3662 ± 712	4471 ± 281	3972 ± 849
HalfCheetah-v2	15254 ± 751	**15701 ± 630**	7396 ± 677	7226 ± 449	12028 ± 191	11769 ± 321
Ant-v2	5532 ± 1266	1163 ± 1326	3659 ± 523	3008 ± 375	**7179 ± 190**	6584 ± 455
Humanoid-v2	**8081 ± 1149**	768 ± 215	5243 ± 200	2738 ± 982	6858 ± 373	5709 ± 1081
Swimmer-v2	114 ± 21	**143 ± 3**	35 ± 8	38 ± 7	69 ± 29	70 ± 40

Clipped Double Q-Learning (Implementation)

(D): with clipped double Q-Learning [Fujimoto et al. 2018]

(S): without clipped double Q-Learning

SAC (D) improves the performance significantly in the termination environment (Hop, Walk, Ant, Humanoid)

	SAC (D)	SAC (S)	AWAC (D)	AWAC (S)	MPO (D)	MPO (S)
Hopper-v2	**3013 ± 602**	1601 ± 733	2329 ± 1020	2540 ± 755	2352 ± 959	2136 ± 1047
Walker2d-v2	**5820 ± 411**	1888 ± 922	3307 ± 780	3662 ± 712	4471 ± 281	3972 ± 849
HalfCheetah-v2	15254 ± 751	**15701 ± 630**	7396 ± 677	7226 ± 449	12028 ± 191	11769 ± 321
Ant-v2	5532 ± 1266	1163 ± 1326	3659 ± 523	3008 ± 375	**7179 ± 190**	6584 ± 455
Humanoid-v2	**8081 ± 1149**	768 ± 215	5243 ± 200	2738 ± 982	6858 ± 373	5709 ± 1081
Swimmer-v2	114 ± 21	**143 ± 3**	35 ± 8	38 ± 7	69 ± 29	70 ± 40

Clipped Double Q-Learning (Implementation)

(D): with clipped double Q-Learning [Fujimoto et al. 2018]

(S): without clipped double Q-Learning

Clipped Double Q doesn't help AWAC or MPO (EM-Controls) as significant as SAC

	SAC (D)	SAC (S)	AWAC (D)	AWAC (S)	MPO (D)	MPO (S)
Hopper-v2	**3013 ± 602**	1601 ± 733	2329 ± 1020	2540 ± 755	2352 ± 959	2136 ± 1047
Walker2d-v2	**5820 ± 411**	1888 ± 922	3307 ± 780	3662 ± 712	4471 ± 281	3972 ± 849
HalfCheetah-v2	15254 ± 751	**15701 ± 630**	7396 ± 677	7226 ± 449	12028 ± 191	11769 ± 321
Ant-v2	5532 ± 1266	1163 ± 1326	3659 ± 523	3008 ± 375	**7179 ± 190**	6584 ± 455
Humanoid-v2	**8081 ± 1149**	768 ± 215	5243 ± 200	2738 ± 982	6858 ± 373	5709 ± 1081
Swimmer-v2	114 ± 21	**143 ± 3**	35 ± 8	38 ± 7	69 ± 29	70 ± 40

Clipped Double Q-Learning (Implementation)

(D): with clipped double Q-Learning [Fujimoto et al. 2018]

(S): without clipped double Q-Learning

➜ Clipped double Q-learning might be a co-dependent and indispensable choice to KL control methods

	SAC (D)	SAC (S)	AWAC (D)	AWAC (S)	MPO (D)	MPO (S)
Hopper-v2	**3013 ± 602**	1601 ± 733	2329 ± 1020	2540 ± 755	2352 ± 959	2136 ± 1047
Walker2d-v2	**5820 ± 411**	1888 ± 922	3307 ± 780	3662 ± 712	4471 ± 281	3972 ± 849
HalfCheetah-v2	15254 ± 751	**15701 ± 630**	7396 ± 677	7226 ± 449	12028 ± 191	11769 ± 321
Ant-v2	5532 ± 1266	1163 ± 1326	3659 ± 523	3008 ± 375	**7179 ± 190**	6584 ± 455
Humanoid-v2	**8081 ± 1149**	768 ± 215	5243 ± 200	2738 ± 982	6858 ± 373	5709 ± 1081
Swimmer-v2	114 ± 21	**143 ± 3**	35 ± 8	38 ± 7	69 ± 29	70 ± 40

Action Distribution for Policy (Implementation)

Tanh-squashed Gaussian Policy ablations

Similar to Clipped double Q-Learning, Tanh-policy seems to **highly co-adapted choice**, and EM-Controls couldn't enjoy large performance gains

For EM-Control, proper action clipping is important to avoid numerical error

$$a \in [-1+\epsilon, 1-\epsilon]^{|\mathcal{A}|}$$

	SAC (w/)	SAC (w/o)	AWR (w/)	AWR (w/o)	MPO (w/)	MPO (w/o)
Hopper-v2	3013 ± 602	6 ± 10	2709 ± 905	$\mathbf{3085 \pm 593}$	2149 ± 849	2136 ± 1047
Walker2d-v2	$\mathbf{5820 \pm 411}$	$-\infty$	3295 ± 335	4717 ± 678	3167 ± 815	3972 ± 849
HalfCheetah-v2	$\mathbf{15254 \pm 751}$	$-\infty$	3653 ± 652	5742 ± 667	9523 ± 312	11769 ± 321
Ant-v2	5532 ± 1266	$-\infty$	445 ± 106	1127 ± 224	2880 ± 306	$\mathbf{6584 \pm 455}$
Humanoid-v2	$\mathbf{8081 \pm 1149}$	$108 \pm 82^{\dagger}$	$2304 \pm 1629^{\dagger}$	5573 ± 1020	6688 ± 192	5709 ± 1081
Swimmer-v2	114 ± 21	28 ± 11	121 ± 3	$\mathbf{128 \pm 4}$	110 ± 42	70 ± 40

Code-level design choices (Code)

While their mathematical connection, these modern inference-based methods have very different code-level detailed choices (referring available open-source code)

Architecture	MPO	AWR	AWAC	SAC
Policy network	(256, 256, 256)	(128, 64)	(256, 256)	(256, 256)
Value network	(512, 512, 256)	(128, 64)	(256, 256)	(256, 256)
Activation function	ELU	ReLU	ReLU	ReLU
Layer normalization	✓	–	–	–
Input normalization	–	✓	–	–
Optimizer	Adam	SGD (momentum=0.9)	Adam	Adam
Learning rate (policy)	1e-4	5e-5	3e-4	3e-4
Learning rate (value)	1e-4	1e-2	3e-4	3e-4
Weight initialization	Uniform	Xavier Uniform	Xavier Uniform	Xavier Uniform
Initial output scale (policy)	1.0	1e-4	1e-2	1e-2
Target update	Hard	–	Soft (5e-3)	Soft (5e-3)
Clipped Double Q	False	–	True	True

Code-level design choices (Code)

While their mathematical connection, these modern inference-based methods have very different code-level detailed choices (referring available open-source code)

Architecture	MPO	AWR	AWAC	SAC
Policy network	(256, 256, 256)	(128, 64)	(256, 256)	(256, 256)
Value network	(512, 512, 256)	(128, 64)	(256, 256)	(256, 256)
Activation function	ELU	ReLU	ReLU	ReLU
Layer normalization	✓	–	–	–
Input normalization		✓		
Learning rate (policy)	1e-4	3e-5	3e-4	3e-4
Learning rate (value)	1e-4	1e-2	3e-4	3e-4
Weight initialization	Uniform	Xavier Uniform	Xavier Uniform	Xavier Uniform
Initial output scale (policy)	1.0	1e-4	1e-2	1e-2
Target update	Hard	–	Soft (5e-3)	Soft (5e-3)
Clipped Double Q	False	–	True	True

Focus on experimental analysis on activation function, normalization, and network size

Activation and Normalization (Code)

ELU and Layer Normalization (MPO's code-level design choices)

	Hopper-v2	Walker2d-v2	HalfCheetah-v2	Ant-v2	Humanoid-v2	Swimmer-v2
SAC	3013 ± 602	$\mathbf{5820 \pm 411}$	15254 ± 751	5532 ± 1266	8081 ± 1149	114 ± 21
SAC-E$^+$	2337 ± 903	5504 ± 431	$\mathbf{15350 \pm 594}$	6457 ± 828	$\mathbf{8196 \pm 892}$	$\mathbf{146 \pm 7}$
SAC-L$^+$	2368 ± 179	5613 ± 762	13074 ± 2218	$\mathbf{7349 \pm 176}$	8146 ± 470	99 ± 18
SAC-E$^+$L$^+$	1926 ± 417	5751 ± 400	12555 ± 1259	7017 ± 132	7687 ± 1385	143 ± 9
MPO	2136 ± 1047	3972 ± 849	11769 ± 321	6584 ± 455	5709 ± 1081	70 ± 40
MPO-E$^-$	2700 ± 879	3553 ± 1145	11638 ± 664	5917 ± 702	4870 ± 1917	108 ± 28
MPO-L$^-$	824 ± 250	2413 ± 1352	6064 ± 4596	2135 ± 2988	5039 ± 838	$-\infty$
MPO-E$^-$L$^-$	843 ± 168	1708 ± 663	-1363 ± 20965	807 ± 2351	5566 ± 787	$-\infty$
AWR	3085 ± 593	4717 ± 678	5742 ± 667	1127 ± 224	5573 ± 1020	128 ± 4
AWR-E$^+$	1793 ± 1305	4418 ± 319	5910 ± 754	2288 ± 715	6708 ± 226	128 ± 4
AWR-L$^+$	2525 ± 1130	4900 ± 671	5391 ± 232	639 ± 68	5962 ± 376	129 ± 2
AWR-E$^+$L$^+$	$\mathbf{3234 \pm 118}$	4906 ± 304	6081 ± 753	2283 ± 927	6041 ± 270	130 ± 1

Activation and Normalization (Code)

ELU and Layer Normalization (MPO's code-level design choices)

	Hopper-v2	Walker2d-v2	HalfCheetah-v2	Ant-v2	Humanoid-v2	Swimmer-v2
SAC	3013 ± 602	$\mathbf{5820 \pm 411}$	15254 ± 751	5532 ± 1266	8081 ± 1149	114 ± 21
SAC-E$^+$	2337 ± 903	5504 ± 431	$\mathbf{15350 \pm 594}$	6457 ± 828	$\mathbf{8196 \pm 892}$	$\mathbf{146 \pm 7}$
SAC-L$^+$	2368 ± 179	5613 ± 762	13074 ± 2218	$\mathbf{7349 \pm 176}$	8146 ± 470	99 ± 18
SAC-E$^+$L$^+$	1926 ± 417	5751 ± 400	12555 ± 1259	7017 ± 132	7687 ± 1385	143 ± 9
MPO	2136 ± 1047	3972 ± 849	11769 ± 321	6584 ± 455	5709 ± 1081	70 ± 40
AWR-E$^+$	1793 ± 1305	4418 ± 319	5910 ± 754	2288 ± 715	6708 ± 226	128 ± 4
AWR-L$^+$	2525 ± 1130	4900 ± 671	5391 ± 232	639 ± 68	5962 ± 376	129 ± 2
AWR-E$^+$L$^+$	$\mathbf{3234 \pm 118}$	4906 ± 304	6081 ± 753	2283 ± 927	6041 ± 270	130 ± 1

SAC + ELU/LayerNorm improves the performance, and beats MPO in Ant

Activation and Normalization (Code)

ELU and Layer Normalization (MPO's code-level design choices)

MPO - ELU/LayerNorm drops the performance significantly
→ ELU/LayerNorm are transferable and beneficial to both EM/KL Control

	Hopper-v2	Walker2d-v2	HalfCheetah-v2	Ant-v2	Humanoid-v2	Swimmer-v2
MPO	2136 ± 1047	3972 ± 849	11769 ± 321	6584 ± 455	5709 ± 1081	70 ± 40
MPO-E$^-$	2700 ± 879	3553 ± 1145	11638 ± 664	5917 ± 702	4870 ± 1917	108 ± 28
MPO-L$^-$	824 ± 250	2413 ± 1352	6064 ± 4596	2135 ± 2988	5039 ± 838	$-\infty$
MPO-E$^-$L$^-$	843 ± 168	1708 ± 663	-1363 ± 20965	807 ± 2351	5566 ± 787	$-\infty$
AWR	3085 ± 593	4717 ± 678	5742 ± 667	1127 ± 224	5573 ± 1020	128 ± 4
AWR-E$^+$	1793 ± 1305	4418 ± 319	5910 ± 754	2288 ± 715	6708 ± 226	128 ± 4
AWR-L$^+$	2525 ± 1130	4900 ± 671	5391 ± 232	639 ± 68	5962 ± 376	129 ± 2
AWR-E$^+$L$^+$	$\mathbf{3234 \pm 118}$	4906 ± 304	6081 ± 753	2283 ± 927	6041 ± 270	130 ± 1

Network Size (Code)

(L): Large size network for MPO; (256, 256, 256) policy & (512, 512, 256) value

(M): Medium size network for SAC; (256, 256) policy & value

(S): Small size network for AWR; (128, 64) policy & value

	Hopper-v2	Walker2d-v2	HalfCheetah-v2	Ant-v2	Humanoid-v2	Swimmer-v2
SAC (L)	2486 ± 746	3188 ± 2115	$\mathbf{16528 \pm 183}$	$\mathbf{7495 \pm 405}$	$\mathbf{8255 \pm 578}$	118 ± 26
SAC (M)	3013 ± 602	$\mathbf{5820 \pm 411}$	15254 ± 751	5532 ± 1266	8081 ± 1149	114 ± 21
SAC (S)	$\mathbf{3456 \pm 81}$	4939 ± 284	12241 ± 400	3290 ± 691	7724 ± 497	59 ± 11
MPO (L)	2136 ± 1047	3972 ± 849	11769 ± 321	6584 ± 455	5709 ± 1081	70 ± 40
MPO (M)	661 ± 79	1965 ± 1426	$-\infty$	5192 ± 538	6015 ± 771	81 ± 28
MPO (S)	430 ± 99	2055 ± 990	5003 ± 1567	3587 ± 957	4745 ± 1428	59 ± 28
AWR (L)	3221 ± 193	4688 ± 648	4360 ± 542	35 ± 43	665 ± 54	$\mathbf{133 \pm 3}$
AWR (M)	2816 ± 910	4826 ± 547	5538 ± 720	413 ± 117	3849 ± 1647	$\mathbf{133 \pm 2}$
AWR (S)	3085 ± 593	4717 ± 678	5742 ± 667	1127 ± 224	5573 ± 1020	128 ± 4

Network Size (Code)

(L): Large size network for MPO; (256, 256, 256) policy & (512, 512, 256) value

(M): Medium size network for SAC; (256, 256) policy & value

(S): Small size network for AWR; (128, 64) policy & value

	Hopper-v2	Walker2d-v2	HalfCheetah-v2	Ant-v2	Humanoid-v2	Swimmer-v2
SAC (L)	2486 ± 746	3188 ± 2115	$\mathbf{16528 \pm 183}$	$\mathbf{7495 \pm 405}$	$\mathbf{8255 \pm 578}$	118 ± 26
SAC (M)	3013 ± 602	$\mathbf{5820 \pm 411}$	15254 ± 751	5532 ± 1266	8081 ± 1149	114 ± 21
SAC (S)	$\mathbf{3456 \pm 81}$	4939 ± 284	12241 ± 400	3290 ± 691	7724 ± 497	59 ± 11
AWR (M)	2816 ± 910	4826 ± 547	5538 ± 720	413 ± 117	3849 ± 1647	$\mathbf{133 \pm 2}$
AWR (S)	3085 ± 593	4717 ± 678	5742 ± 667	1127 ± 224	5573 ± 1020	128 ± 4

SAC is robust to network size
(L) and (M) can be first choices

Network Size (Code)

(L): Large size network for MPO; (256, 256, 256) policy & (512, 512, 256) value

(M): Medium size network for SAC; (256, 256) policy & value

(S): Small size network for AWR; (128, 64) policy & value

MPO highly depends on Large size network

MPO (L)	2136 ± 1047	3972 ± 849	11769 ± 321	6584 ± 455	5709 ± 1081	70 ± 40
MPO (M)	661 ± 79	1965 ± 1426	$-\infty$	5192 ± 538	6015 ± 771	81 ± 28
MPO (S)	430 ± 99	2055 ± 990	5003 ± 1567	3587 ± 957	4745 ± 1428	59 ± 28
AWR (L)	3221 ± 193	4688 ± 648	4360 ± 542	35 ± 43	665 ± 54	$\mathbf{133 \pm 3}$
AWR (M)	2816 ± 910	4826 ± 547	5538 ± 720	413 ± 117	3849 ± 1647	$\mathbf{133 \pm 2}$
AWR (S)	3085 ± 593	4717 ± 678	5742 ± 667	1127 ± 224	5573 ± 1020	128 ± 4

Network Size (Code)

(L): Large size network for MPO; (256, 256, 256) policy & (512, 512, 256) value

(M): Medium size network for SAC; (256, 256) policy & value

(S): Small size network for AWR; (128, 64) policy & value

	Hopper-v2	Walker2d-v2	HalfCheetah-v2	Ant-v2	Humanoid-v2	Swimmer-v2
SAC (L)	2486 ± 746	3188 ± 2115	$\mathbf{16528 \pm 183}$	$\mathbf{7495 \pm 405}$	$\mathbf{8255 \pm 578}$	118 ± 26
SAC (M)	3013 ± 699	$\mathbf{5820 \pm 411}$	15954 ± 751	5538 ± 1966	8081 ± 1140	114 ± 91
MPO (S)	430 ± 99	2055 ± 990	5003 ± 1567	3587 ± 957	4745 ± 1428	59 ± 28
AWR (L)	3221 ± 193	4688 ± 648	4360 ± 542	35 ± 43	665 ± 54	$\mathbf{133 \pm 3}$
AWR (M)	2816 ± 910	4826 ± 547	5538 ± 720	413 ± 117	3849 ± 1647	$\mathbf{133 \pm 2}$
AWR (S)	3085 ± 593	4717 ± 678	5742 ± 667	1127 ± 224	5573 ± 1020	128 ± 4

AWR shows the robustness in low-dim state tasks,
while flagilities in high-dim state tasks (Ant 111-dim, Humanoid 376-dim)

Summary

1. We first reformulated recent **inference-based off-policy algorithms** (MPO, AWR, SAC) into a **unified mathematical objective**, and exhaustively clarified the algorithmic and implementational differences
2. Though the extensive and precise ablations, we found the co-adaptive and no-co-adaptive implementation & code details
3. EM-Control seems robust to implementation, while KL-Control seems robust to code design choices

We hope our work can encourage more works that study precisely the impacts of algorithmic properties and empirical design choices across a broader spectrum of deep RL algorithms