MOOSS: Mask-Enhanced Temporal Contrastive Learning for Smooth State Evolution in Visual Reinforcement Learning – Supplementary Material –

Jiarui Sun, M. Ugur Akcal, Girish Chowdhary
University of Illinois Urbana-Champaign
Urbana, IL, USA
{jsun57, makcal2, girishc}@illinois.edu

Wei Zhang
Visa Research
Foster City, CA, USA
wzhan@visa.com

A. Additional Backgrounds

A.1. Soft Actor Critic

Soft Actor-Critic (SAC) [3] is an off-policy, model-free actor-critic Reinforcement Learning (RL) algorithm that follows the entropy-regularized RL framework. This framework introduces the concept of entropy into the RL objective to encourage exploration. In particular, SAC tries to learn (1) a soft Q-function $Q_\omega(\cdot)$, (2) a soft state value function $V_\psi(\cdot)$, and (3) a policy $\pi_\eta(\cdot)$. Let $s_t \in \mathcal{S}$ denote the state at timestep t. $V_\psi(\cdot)$ is trained to minimize the MSE:

$$J_V(\psi) = \mathbb{E}_{s_t\sim\mathcal{D}}[\frac{1}{2}(V_\psi(s_t) - \mathbb{E}[Q_w(s_t,a_t) - \log\pi_\eta(a_t|s_t)])^2], \tag{A.1}$$

where $\mathcal{D}$ is the replay buffer. $Q_\omega(\cdot)$ is trained to minimize the soft Bellman residual:

$$J_Q(\omega) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}[\frac{1}{2}(Q_\omega(s_t,a_t) - (r_t + \gamma\mathbb{E}_{s_{t+1}\sim\rho_\pi(s)}[V_{\bar{\psi}}(s_{t+1})]))^2], \tag{A.2}$$

where $\rho_\pi(s)$ denotes state marginal of the state distribution induced by π, and $V_{\bar{\psi}}$'s parameters $\bar{\psi}$ are updated by the Exponential Moving Average (EMA) of ψ (or only gets updated periodically) for training stability. Policy π is optimized to maximize the expected return and the entropy at the same time:

$$J_\pi(\eta) = \mathbb{E}_{s_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}[\log\pi_\eta(f_{\pi_\eta}(\epsilon_t;s_t)|s_t) - Q(s_t, f_{\pi_\eta}(\epsilon_t;s_t))], \tag{A.3}$$

where ϵ_t is the input noise vector sampled from a standard Gaussian $\mathcal{N}$, and $f_{\pi_\eta}(\epsilon_t;s_t)$ denotes actions sampled stochastically from $\pi_\eta(\cdot)$. This sampling procedure is accomplished via the reparameterization trick proposed in [6]. Given its performance, SAC serves as a robust baseline for continuous control tasks.

A.2. Deep Q-Network and Rainbow

Deep Q-Network (DQN) [7] is the first deep RL algorithm that successfully learns control policies directly from visual data, *i.e.*, image-based observations. Facilitated by deep neural networks, it greatly improves the training procedure of Q-learning by using (1) an experience replay buffer for drawing samples and (2) a target Q-network $Q_{\omega'}(\cdot)$ to stabilize training. $Q_{\omega'}(\cdot)$ shares the same architecture with the Q-network $Q_\omega(\cdot)$ and is kept frozen as the optimization target every C steps, where C is a hyperparameter. $Q_\omega(\cdot)$ is trained to minimize the mean square error:

$$J_Q(\omega) = \mathbb{E}_{(s_t,a_t,s_t)\sim\mathcal{D}}[Q_\omega(s_t,a_t) - (r_t + \gamma\max_a Q_{\omega'}(s_{t+1},a))^2]. \tag{A.4}$$

Rainbow [4] is an enhanced DQN variant that amalgamates multiple advancements into a unified RL agent, featuring (1) double DQN [10], (2) prioritized experience replay [8], (3) dueling networks [12], (4) multi-step return [9], (5) distributional RL as in [1], and (6) noisy layers [2]. By integrating these techniques, Rainbow is considered a robust baseline for discrete control tasks.

B. MOOSS Implementation Details

B.1. Network Architecture

MOOSS-equipped RL framework consists of two parts: (1) Modules that are necessary for the RL algorithms (SAC and Rainbow), such as the Q-network $Q_\omega(\cdot)$ and the observation encoder $f_\theta(\cdot)$; (2) Additional modules required by MOOSS, *i.e.*, the predictive decoder $g_\phi(\cdot)$.

For the first part, we mainly adopt the implementations of SAC and Rainbow from [13] for fair comparisons. Specifically, the observation encoder $f_\theta(\cdot)$ in SAC is built from 4 convolutional layers with ReLU activations, followed by a projection through a linear layer and normal-

Steps	Model	Reacher, hard	Walker, run
100k	*Base*	341 ± 275	105 ± 47
100k	MOOSS	**779 ± 379**	**164 ± 6**
500k	*Base*	669 ± 290	466 ± 39
500k	MOOSS	**980 ± 11**	**509 ± 25**

Table A.1. Results on harder DMC tasks.

ization. Note that we use a state representation dimension $d = 64$ instead of 50 to allow multi-head attention on $g_\phi(\cdot)$. On the other hand, in Rainbow, $f_\theta(\cdot)$ includes 3 convolutional layers with ReLU activations, while the Q-learning heads utilize a multilayer perceptron (MLP) design. These observation encoders correspond to the query encoder depicted in Fig. 1 of the main paper, and the key encoder $f_{\bar{\theta}}(\cdot)$ adopts the identical architecture as $f_\theta(\cdot)$.

The additional predictive decoder $g_\phi(\cdot)$, necessary for MOOSS, comprises 2 transformer encoder layers, each with 4 attention heads. The causality of $g_\phi(\cdot)$ is enforced using a causal attention mask. Actions a_t are converted into action embeddings $\mathbf{a}_t \in \mathbb{R}^d$ via a linear layer, and the positional encodings employed are the standard absolute sinusoidal positional encodings introduced in [11].

B.2. General Learning Settings

We mainly follow the training pipeline of [13] to train MOOSS. As such, Adam [5] is used to optimize all trainable parameters, and MOOSS is trained until reaching the designated maximum agent-environment interaction steps. The hyper-parameters for DMC and Atari are listed in Tab. A.3 and Tab. A.4, respectively, with the **bolded** ones being tuned for performance analysis. Notably, in Atari, few games employ a masking ratio of $p_m = 10\%$ and a temporal window size of $L = 2$ to enhance game performance. These games typically feature small, fast-moving objects crucial to success. For instance, *Pong* includes a small ping-pong ball crucial for scoring points, while *Gopher* challenges players to stop fast-moving gophers from eating carrots. As discussed in the main paper, for games with fast-moving objects, the high masking ratio of $p_m = 50\%$ can lead to excessive information loss, while an overly long contrastive window, with $L = 6$, may become counterproductive. This suggests that a large temporal window might encompass states that are too similar, diminishing the effectiveness of MOOSS in these scenarios.

C. Additional Experiments

C.1. Performance on Harder Tasks from DMC

In Tab. A.1, we extend our analysis by comparing MOOSS with its *Base* model on two challenging tasks from DMC: *Reacher-hard* and *Walker-run*. These tasks have not been previously utilized to evaluate the sample efficiency of visual RL algorithms. The results reveal that MOOSS consistently enhances the performance on these difficult tasks compared to the *Base* variant, underscoring our method's effectiveness. Notably, the performance improvements are more pronounced at 100k steps, which is the low data regime. This further highlights the benefits of modeling the smooth evolution of states on sample efficiency.

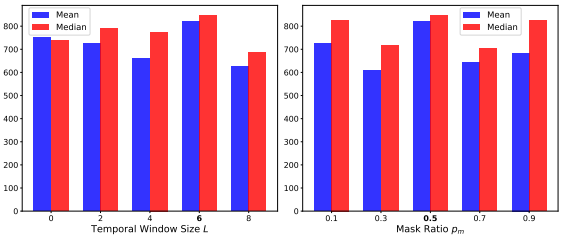

Figure A.1. Ablation on window size L and masking ratio p_m.

Depth	$g_\phi(\cdot)$ Size	Mean	Median
1	63.27K	660.1	690.0
2 (ours)	113.25K	**818.6**	**847.5**
3	163.24K	695.8	753.5
4	213.22K	667.9	847.0

Table A.2. Ablation on decoder depth.

C.2. Temporal Window Size and Masking Ratio

In this section, we examine how MOOSS's key hyper-parameters, *i.e.*, temporal window size L and masking ratio p_m, affect its performance. The results in Fig. A.1 on temporal window size present a trend where performance initially fluctuates mildly, reaching a peak, and then deteriorates as the window size expands. This trend suggests that the context provided by an overly large temporal window can be counterproductive. We argue that in the case of a large L, for tasks involving repetitive actions (such as *Walker*), states that are temporally distant may also appear similar, leading to confusion and diminishing MOOSS's performance. We also find that $p_m = 50\%$ is a proper choice for MOOSS. This choice strikes a balance between challenging MOOSS to exploit spatial-temporal correlations across observations for query generation, and retaining enough unmasked content to facilitate meaningful learning. Such level of masking properly ensures that MOOSS is neither overwhelmed by excessive information loss nor under-stimulated by an abundance of visible data.

C.3. Ablation on Decoder Depth

In Tab. A.2, we study the effect of numbers of Transformer layers used in the decoder. We observe that the depth of $g_\phi(\cdot)$ is pivotal to MOOSS's performance, with 2 emerging as the optimal choice. The result underscores the necessity of a decoder with balanced power in MOOSS; it must be sufficiently effective in reducing possible ambiguities in masked state embeddings, but not so dominant as to usurp the learning role of the observation encoder.

D. Discussion on Limitations

While effective, MOOSS's performance gain on Atari is relatively lower compared to DMC. Delving into this, we observe that MOOSS does not perform as well in Atari games featuring small, fast-moving objects crucial to success, like bullets. This is particularly evident in games such as Battle Zone, compared to its performance in other games. This may be because MOOSS's temporal contrastive objective becomes less effective in capturing drastic key changes across states, and is further challenged by spatial-temporal masking, which might result in excessive information loss. Besides, MOOSS requires hyper-parameters that may need additional tuning for different applications.

Additionally, we recognize that certain tasks may violate MOOSS's "gradually evolving state" assumption, as discussed in the Limitation Section. However, we first note that in scenarios with frequent background changes (*e.g.*, Hero from Atari), MOOSS proves *advantageous* as it guides the encoder to filter out task-irrelevant background information, thereby focusing on task-essential elements. Second, while MOOSS does not inherently address fast moving agents algorithmically, this issue is mitigated by the action_repeat hyperparameter in RL algorithms. action_repeat is usually adjusted to a small value for environments with rapid observation/agent changes (*e.g.*, 2 for Spin vs. 8 for Swing from DMControl), to stabilizes temporal state dynamics and thus enhances RL model performance. In MOOSS, action_repeat is not specifically tuned. Thus, given MOOSS's benefit from this mechanism, violations of gradual state evolution assumption are likely rare.

References

[1] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017. 1

[2] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017. 1

[3] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 1

[4] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 1

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[6] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 1

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 1

[8] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015. 1

[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 1

[10] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. 1

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[12] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016. 1

[13] Tao Yu, Zhizheng Zhang, Cuiling Lan, Yan Lu, and Zhibo Chen. Mask-based latent reconstruction for reinforcement learning. *Advances in Neural Information Processing Systems*, 35:25117–25131, 2022. 1, 2

Hyper-parameter	Value
Frame stack ($c/3$)	3
Observation rendering	$(100, 100)$
Observation downsampling ($H \times W$)	$(84, 84)$
Augmentation	Random crop and random intensity
Replay buffer size	100000
Initial exploration steps	1000
Action repeat	2 *Finger-spin* and *Walker-walk*; 8 *Cartpole-swingup*; 4 otherwise
Evaluation episodes	10
Optimizer	Adam
(β_1, β_2) (Except α)	$(0.9, 0.999)$
$(\beta_1, \beta_2) \rightarrow (\alpha)$ (temperature in SAC)	$(0.5, 0.999)$
Learning rate for base RL modules	0.0002 *Cheetah-run*; 0.001 otherwise
Learning rate for MOOSS-specific modules	0.0001 *Cheetah-run*; 0.0005 otherwise
Learning rate warmup for MOOSS-specific modules	6000 steps
Learning rate	0.0001
Batch size for policy learning	512
Batch size for auxiliary task	128
Q-function EMA m	0.99
Critic target update frequency	2
Discount factor	0.99
Initial temperature	0.1
Target network update period	1
Target network EMA m	0.9 *Walker-walk*; 0.95 otherwise
State representation dimension d	64
MOOSS Specific Hyper-parameters	
Weight of MOOSS loss λ	0.1
Sequence length F	16
Cube spatial size $h \times w$	7×7
Cube temporal length f	4 *Cartpole-swingup* and *Reacher-easy* 8 otherwise
Initial Contrastive temperature τ_0	0.07
Contrastive temperature skip $\tau_{l+1} - \tau_l$	0.075
Predictive decoder $\mathbf{g}_\phi(\cdot)$ depth	2
Random walk mask ratio $\mathbf{p_m}$	50%
Temporal window size L	6

Table A.3. Hyper-parameters used for DMC.

Hyper-parameter	Value
Gray-scaling	True
Frame stack ($c/3$)	4
Observation downsampling ($H \times W$)	$(84, 84)$
Augmentation	Random crop and random intensity
Action repeat	4
Training steps	100k
Max frames per episode	108k
Reply buffer size	100k
Minimum replay size for sampling	2000
Mini-batch size	32
Optimizer, (learning rate, $\beta_1, \beta_2, \epsilon$)	Adam, (0.0001, 0.9, 0.999, 0.00015)
Max gradient norm	10
Update	Distributional Q
Dueling	True
Support of Q-distribution	51 bins
Discount factor	0.99
Reward clipping Frame stack	$[-1, 1]$
Priority exponent, correction	0.5, $0.4 \rightarrow 1$
Exploration	Noisy nets
Noisy nets parameter	0.5
Evaluation trajectories	100
Replay period every	1 step
Updates per step	2
Multi-step return length	10
Q-network: channels	$32, 64, 64$
Q-network: filter size	$8 \times 8, 4 \times 4, 3 \times 3$
Q-network: stride	$4, 2, 1$
Q-network: hidden units	256
Target network update period	1
EMA coefficient m	0
MOOSS Specific Hyper-parameters	
Weight of MOOSS loss λ	0.1
Sequence length F	16
Cube spatial size $h \times w$	7×7
Cube temporal length f	4
Initial Contrastive temperature τ_0	0.07
Contrastive temperature skip $\tau_{l+1} - \tau_l$	0.075
Predictive decoder $g_\phi(\cdot)$ depth	2
Random walk mask ratio $\mathbf{p_m}$	10% *Gopher*, *Kangaroo*, *Ms Pacman*, *Pong*, *Seaquest* 50% otherwise
Temporal window size L	2 *Gopher*, *Kangaroo*, *Ms Pacman*, *Pong*, *Seaquest* 6 otherwise

Table A.4. Hyper-parameters used for Atari.