Multi-task Self-Supervised Adaptation for Reinforcement Learning

Keyu Wu[1], Zhenghua Chen[1*], Min Wu[1], Shili Xiang[1], Ruibing Jin[1], Le Zhang[2], and Xiaoli Li[1]
[1]*Institute for Infocomm Research, A*STAR, Singapore*
[2]*University of Electronic Science and Technology of China, China*
wu_keyu@i2r.a-star.edu.sg, chen0832@e.ntu.edu.sg

***Abstract*—Policy adaptation remains one of the key challenges for reinforcement learning (RL). Thus, RL agents often fail to generalize to unseen scenarios. In this paper, we propose to improve the generalization of RL algorithms through multi-task self-supervised adaptation (MSSA). The proposed method is a general paradigm that can be implemented on top of any RL algorithm. It better extracts high-level feature representations from augmented observations through incorporating multiple self-supervised learning tasks with complementary objectives. The selected self-supervision tasks include rotation prediction, inverse dynamics prediction and contrastive learning. It then performs control actions based on the extracted features. The proposed MSSA method consistently outperforms all the baseline methods on diverse complex tasks in the DeepMind Control suite benchmark and sets new state-of-the-art results without incurring longer inference time. It is demonstrated that MSSA has superior generalization capability and is robust to environmental changes.**

***Index Terms*—reinforcement learning, policy adaptation, policy generalization, self-supervised learning**

I. Introduction

Without any labeled data, reinforcement learning (RL) can learn from previous experiences automatically and hence has achieved prominent success in a wide range of areas, such as video games [1], robotics [2]–[5], data selection [6] and so on. Currently, however, vision-based RL remains plagued by its poor generalization ability. That is, RL agents trained in one environment can hardly generalize to unseen scenarios typically [7]. Due to the partially-observable and high-dimensional nature of pixel inputs, the policy adaptation problem of RL can be further exacerbated in vision-based tasks. As a result, the deployment of vision-based RL is significantly limited in real-world applications.

One well explored solution is domain randomization, which typically creates a variety of environments with random properties so as to figure out the best policy that can work across all the environments [8]. Nonetheless, domain randomization is sensitive to the choice of randomized parameters and can also lead to large model complexity. Another solution is domain adaptation, which aims to mitigate the distribution discrepancy between the source and target domains [7], [9]. Nevertheless, in domain adaptation, the target domain data are assumed to be accessible during training while RL agents are typically required to be deployed in completely unknown environments. In addition, data augmentation is also demonstrated to be effective to improve the RL generalization ability recently [10]–[12].

Instead of learning a generalizable policy that is robust to all possible environmental changes, an unsupervised policy adaptation method, Policy Adaptation during Deployment (PAD), is proposed in [13], which explores the use of single self-supervision task to enable continued training during deployment and improve the generalization of vision-based RL. However, the online fine-tuning of PAD can result in adaptation delay, higher memory requirement as well as longer inference time.

In this paper, we propose a Multi-task Self-Supervised Adaptation (MSSA) method to address the generalization challenge for RL. MSSA jointly trains the RL policy with multiple self-supervised objectives. The selected auxiliary self-supervision tasks, i.e., rotation prediction [14], Contrastive Unsupervised Representations for Reinforcement Learning (CURL) [15] and inverse dynamics prediction [13], have different and complementary objectives. Specifically, these three tasks aim to improve the feature representation extraction via better scene understanding, sample efficiency and connection between observations and actions, respectively. Moreover, instead of continuing the learning during deployment, we implement data augmentations on input observations to further improve the generalization capability of the RL models without adaptation delay. It is worth mentioning that MSSA is a general paradigm that can be implemented on top of any RL algorithm and requires almost no change to the underlying algorithms. We evaluate our method on the DeepMind Control suite benchmark [16] through deploying it in new environments with changes that are unknown during training. Experimental results demonstrate the superiority of MSSA which outperforms the baselines and improves generalization in all test environments.

To summarize, the main contributions of our work are:

- We have proposed a Multi-task Self-Supervised Adaptation (MSSA) method to address the generalization challenge for RL. To the best of our knowledge, this is the first work that combines RL with multiple self-supervision tasks.

* Corresponding Author

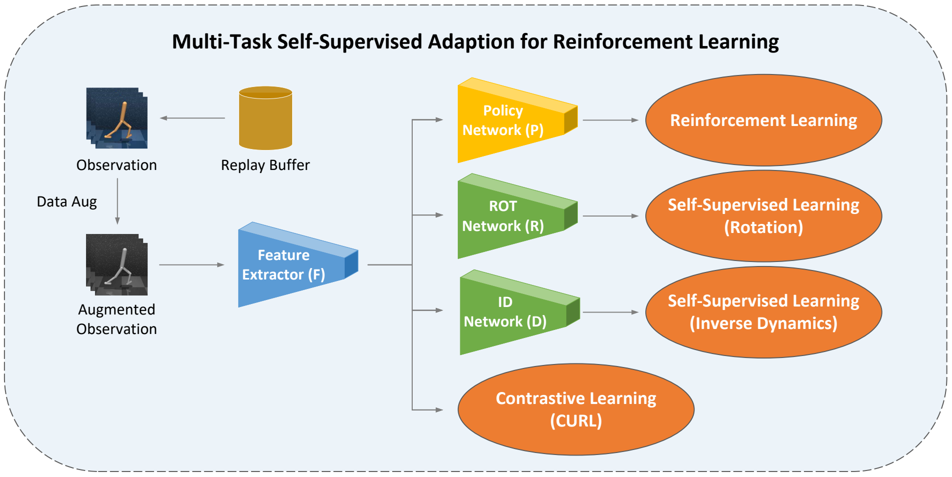

Multi-Task Self-Supervised Adaption for Reinforcement Learning
Observation
Replay Buffer
Data Aug
Augmented Observation
Feature Extractor (F)
Policy Network (P)
ROT Network (R)
ID Network (D)
Reinforcement Learning
Self-Supervised Learning (Rotation)
Self-Supervised Learning (Inverse Dynamics)
Contrastive Learning (CURL)

Fig. 1: Architecture of MSSA. The policy is jointly trained with one RL objective and multiple self-supervised objectives. During training, a batch of transitions is sampled from the replay buffer and data augmentations are applied to the observations before feeding them into the network. The overall network consists of four parts, i.e., the feature extractor F, the policy network P, the rotation prediction network R, and the inverse dynamics prediction network D.

- MSSA significantly improves generalization capability without adaptation delay through combining multiple auxiliary self-supervised learning tasks with complementary objectives and data augmentations. Moreover, MSSA is a general paradigm that can be implemented on top of any RL algorithm without change to the underlying algorithms.
- We show that MSSA outperforms the state-of-the-art baselines on the DeepMind Control suite benchmark, which is widely used to evaluate the generalization capability of RL, by a large margin.

II. Method

In this section, we will introduce the proposed Multi-task Self-Supervised Adaptation (MSSA) approach. Since MSSA minimally modifies a base RL agent, it can be implemented on top of any RL algorithm, both on-policy and off-policy, that target to minimize the RL objective with respect to the network parameters through stochastic gradient descent. In this work, we train MSSA alongside the Soft Actor-Critic (SAC) algorithm [17] to demonstrate its effectiveness.

A. Architecture Overview

The network architecture of MSSA is depicted in Figure 1. Generally, MSSA trains the RL network jointly with multiple auxiliary self-supervised learning networks with complementary objectives, which are for improved scene understanding, data efficiency and connection between observations and actions, respectively. During training, observations are sampled from the replay buffer and are then augmented within the batch. Since it is common in the RL setting to stack consecutive frames as observations to infer temporal information, we apply augmentations randomly across the batch yet consistently across the frame stack to retain the temporal information. Besides, we apply both the random crop and grayscale augmentations simultaneously considering the computation efficiency and the significance of combining random crop with color distortion [10], [18]. The random crop augmentation randomly crops a square patch from the original frame while the grayscale augmentation converts the original RGB images to grayscale images with a probability of p.

The network architecture is modified so that the RL network and the self-supervised learning networks can share the intermediate features. The overall network is split into four parts, i.e., the feature extractor F with parameters θ_f, the policy network P with parameters θ_π, the rotation prediction head R with parameters θ_r, and the inverse dynamics prediction head D with parameters θ_d. Specifically, the feature extractor aims to extract feature representations from the input augmented observations and its parameters are shared by the RL network and all the self-supervised learning networks. As shown in Figure 2, based on the extracted features, the policy network is used to map the representations to actions.

In the meantime, the feature representations of o_t is fed into the rotation network to predict the rotation angle as illustrated in Figure 3, and the feature representations of o_t and o_{t+1} are both passed to the inverse dynamics network to predict the action as depicted Figure 4. In the CURL task which is demonstrated in Figure 5, the anchor and positive observations are the same stack of frames with different data augmentations while the negative ones are from other stacks of frames. The query and key observations are then fed into the query encoder and key encoder, respectively, where the query encoder is the feature extractor F while the key encoder is the momentum-based moving average of F.

B. Self-Supervised Tasks

In this paper, the generalization capability is improved through three different auxiliary self-supervised learning tasks, i.e., rotation prediction task, inverse dynamics task and CURL.

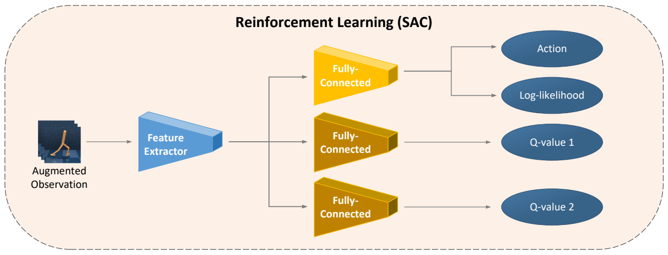


Fig. 2: Architecture of the reinforcement learning network.

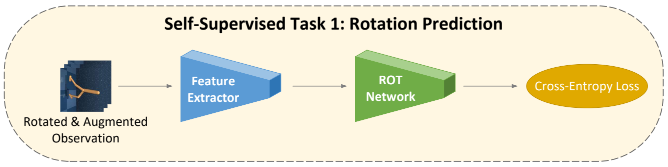

Self-Supervised Task 1: Rotation Prediction
Rotated & Augmented Observation
Feature Extractor
ROT Network
Cross-Entropy Loss

Fig. 3: Description of the rotation prediction task.

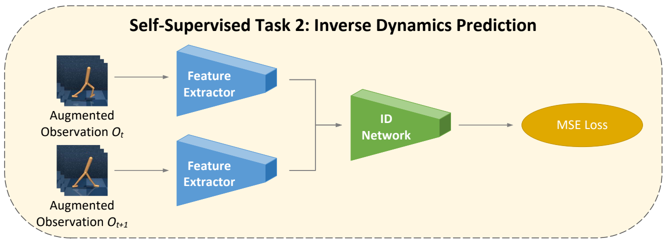

Self-Supervised Task 2: Inverse Dynamics Prediction
Augmented Observation O_t
Augmented Observation O_{t+1}
Feature Extractor
Feature Extractor
ID Network
MSE Loss

Fig. 4: Description of the inverse dynamics prediction task.

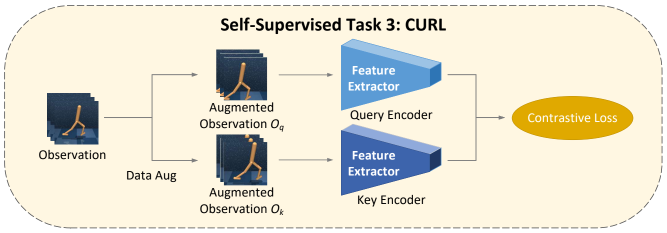


Fig. 5: Description of the CURL task.

The three tasks are capable of deriving a better feature extractor through improved scene understanding, improved connection between observations and actions, and improved data-efficiency, respectively.

Rotation Prediction In the rotation prediction task, we rotate a stack of frames by either 0, 90, 180 or 270 degrees so that the rotation prediction task can be modeled as a four-way classification problem. Formally, if $Rot(\cdot)$ is the rotation operation, then with a batch of input observations $\{\mathbf{o}_t^i\}_{i=1}^{n_b}$ and their corresponding rotation labels $\{\mathbf{y}_t^i\}_{i=1}^{n_b}$, where n_b and t denote the batch size and time step respectively, the cross entropy loss function for the rotation prediction task can be expressed as:

$$L_r = -\frac{1}{n_b}\sum_{i=1}^{n_b} \log\left[\sigma_{\mathbf{y}_t^i}(R(F(Rot(\mathbf{o}_t^i))))\right], \tag{1}$$

where σ denotes the softmax function defined as:

$$\sigma_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad \text{for } i = 1, \dots, k. \tag{2}$$

Inverse Dynamics Prediction The inverse dynamics prediction model takes the two consecutive observations, o_t and o_{t+1}, from the transition (o_t, a_t, o_{t+1}) as its inputs, and predicts the action in between. Therefore, the inverse dynamics prediction task can be modeled as a regression problem with the mean squared error loss function defined as:

$$L_d = \frac{1}{n_b}\sum_{i=1}^{n_b} \frac{1}{2}\left(a_t^i - D(F(\mathbf{o}_t^i), F(\mathbf{o}_{t+1}^i))\right)^2. \tag{3}$$

CURL Using contrastive unsupervised learning, CURL is able to learn rich representations of high-dimensional data. As introduced above, the query observation o_q and key observation o_k need to be generated using different augmentations. Then, the positive observation o_{k_+} of query observation o_q is defined as the same stack of frames with different augmentations while the negatives are defined as the rest stacks of frames in the batch with different augmentations. To measure the agreement between query-key pairs, CURL employs the bi-linear inner-product $F(o_q)^T W F(o_k)$, where W is a learned parameter matrix. By adopting the InfoNCE loss, the contrastive loss function in CURL can be derived as:

$$L_c = -\frac{1}{n_b}\sum_{i=1}^{n_b} \log \frac{\exp(F^T(\mathbf{o}_q^i) W F(\mathbf{o}_{k_+}^i))}{\sum_{j=1}^{n_b} \exp(F^T(\mathbf{o}_q^i) W F(\mathbf{o}_k^j))}. \tag{4}$$

C. Training Framework

In this paper, we adopt the SAC, a state-of-the-art off-policy method, as the base RL algorithm. In SAC, the policy is trained through a maximum entropy framework with the following objective function:

$$J(\pi) = \mathop{\mathbb{E}}_{\tau\sim\rho_\pi}\left[\sum_{t=0}^{T}\gamma^t\Big(r_t + \alpha H(\pi(\cdot|o_t))\Big)\right], \tag{5}$$

where γ is the discount factor, $H(\pi(\cdot|o_t))$ denotes the policy entropy and α is a temperature parameter to determine the significance of the entropy term against the reward term.

As shown in Figure 2, besides the feature extractor, SAC learns a policy π with parameters ϕ and two soft Q-functions with parameters ψ_1, ψ_2 concurrently. Specifically, the critic parameters are updated by minimizing the squared Bellman error:

$$\begin{aligned} L_Q(\psi_i) = \mathbb{E}_{(o_t,a_t,o_{t+1})\sim\mathcal{D}}\Bigg[\frac{1}{2}\Big(&Q_{\psi_i}(o_t,a_t) \\ &- \big(r_t + \mathcal{T}\big)\Big)^2\Bigg], \end{aligned} \tag{6}$$

where $\mathcal{D}$ denotes the replay buffer and $\mathcal{T}$ is the target value which can be computed by:

$$\mathcal{T} = \min_{i=1,2} Q_{\psi_i}(o_{t+1}, \tilde{a}) - \alpha \log\pi_\phi(\tilde{a}|o_{t+1}), \tag{7}$$

TABLE I: Hyperparameters used for the DMC experiments.

Hyperparameter	**Value**
Data Augmentation	random crop grayscale
Observation rendering	100×100
Observation downsampling	84×84
Stacked frames	3
Action repeat	2 (finger) 8 (cartpole) 4 (otherwise)
Discount factor γ	0.99
Episode length	1,000
Base RL algorithm	Soft Actor-Critic
Self-supervised tasks	Rotation Prediction Inverse Dynamics Prediction CURL
Number of training step	100,000
Replay buffer size	100,000
Initial steps	1000
Optimizer (F, P, R, D)	Adam (β_1 = 0.9, β_2 = 0.999)
Optimizer (α)	Adam (β_1 = 0.5, β_2 = 0.999)
Learning rate (F, P, R, D)	1e-3
Learning rate (α)	1e-4
Initial temperature	0.1
Batch size	128
Update frequency	2
Trade-off factors (λ_1, λ_2, λ_3)	1e-3

where $\tilde{a}$ is sampled from policy π_ϕ. Meanwhile, the parameters of π_ϕ can be updated through minimizing the divergence from the exponential of the soft-Q value:

$$L_\pi(\phi) = \mathbb{E}_{o_t \sim \mathcal{D}, \tilde{a} \sim \pi_\phi}\left[\alpha \log \pi_\phi(\tilde{a}|o_t) - \min_{i=1,2} Q_{\psi_i}(o_t, \tilde{a})\right]. \tag{8}$$

During training, the feature extractor is updated without the actor loss. Therefore, the training of MSSA can be summarized as the following optimization problem:

$$\begin{gathered} \min_F L_Q + \lambda_1 L_r + \lambda_2 L_d + \lambda_3 L_c, \\ \min_R L_r, \\ \min_D L_d, \\ \min_P L_Q + L_\pi, \end{gathered} \tag{9}$$

where λ_1, λ_2, λ_3 are trade-off factors.

III. Experiments

A. Setup

DMControl [16] a widely used benchmark dataset for RL algorithms, contains a variety of continuous control tasks. In this work, we evaluate RL agents on eight different tasks where the agents can only observe raw pixels.

For benchmarking, we compare the proposed MSSA method with three different baselines, i.e., SAC [17], RAD [10] and PAD [13]. All methods are implemented on top of SAC. Specifically, RAD achieves policy adaptation via data augmentations while PAD achieves policy adaptation via adding a single self-supervised objective as well as continuing the learning during deployment. In our experiments, RAD adopts the same data augmentations (random crop and grayscale augmentations) as MSSA, and PAD employs the inverse dynamics prediction task for self-supervision as stated in [13].

All RL agents are trained in a fixed environment as shown in Figure 6(a) and are evaluated in unseen test environments. Particularly, the test environments 'color_easy' illustrated in Figure 6(b) and 'color_hard' illustrated in Figure 6(c) randomize the colors of background, foreground and the agent itself, while the test environment 'video' replaces the background with natural videos. Different methods are all evaluated at 100k environment steps and the hyperparameters are listed in Table I. For each test environment, the methods are evaluated across three random seeds and 50 randomly initialized episodes per seed.

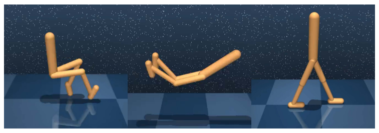
In MSSA, all networks contain 11 convolutional layers with 32 filters followed by 4 fully-connected layers. Specifically, the rotation and inverse dynamics prediction networks share all 11 convolutional layers. Meanwhile, the RL and self-supervised learning networks share the first 8 convolutional layers. The network structures of the baseline methods are consistent with the architecture of MSSA.

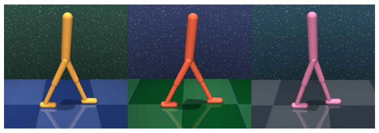
B. Results and Discussions

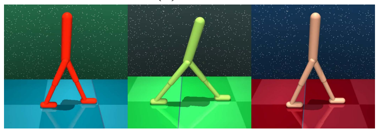
The experimental results obtained in the training environment are shown in Table II, where the best result for each task is highlighted in bold. In addition to the original MSSA method, we also evaluated its four variations. The first three variations adopt different combinations of the self-supervised tasks to explore the best performing combinations in different tasks. Specifically, MSSA (ID+CURL), MSSA (ROT+CURL) and MSSA (ROT+ID) denote variations trained without the rotation prediction task, inverse dynamics prediction task and CURL task, respectively. In the last variation MSSA (w/o Gray Aug), the grayscale augmentation is not applied while the random crop augmentation is retained since it is also applied in baseline methods. For better visualization, we highlight the results of the MSSA variations in red if they are better than the results of all the baseline methods. It is observed that the introduction of multiple self-supervised tasks can also improve the performance in the training environment. Although the highest average reward is achieved by MSSA (w/o Gray Aug), MSSA is more robust as it consistently outperforms all the state-of-the-art methods in all tasks.

The experimental results obtained in the 'color_easy', 'color_hard' and 'video' environments are illustrated in Table III, Table IV and Table V, respectively. Generally, MSSA is much more superior compared to the baseline methods. In particular, both MSSA and MSSA (ID+CURL) outperform the existing methods in all the environments with randomized colors. On the 'video' benchmark, although MSSA demonstrates improvements over baselines, there is still large room for improvement. A possible solution is to improve the dispersing of task-irrelevant visual features.

Overall, some of the most important observations from experimental results are as follows. First, since MSSA outperforms PAD even without the online learning, it is proved that


(a) train


(b) color_easy


(c) color_hard

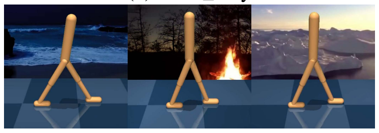

(d) video

Fig. 6: Sample environments for DeepMind Control tasks. The agents are trained in a fixed training environment and evaluated in different test environments.

TABLE II: Cumulative reward obtained in the training environment. The highest scores are highlighted in bold and the results of the MSSA variations are marked in red if they are better than those of all the baselines.

Env	SAC	RAD	PAD	MSSA (ROT+ID+CURL)	MSSA (ID+CURL)	MSSA (ROT+CURL)	MSSA (ROT+ID)	MSSA (w/o Gray Aug)
Walker, walk	616 $\pm$ 6	562 $\pm$ 131	620 $\pm$ 76	657 $\pm$ 70	716 $\pm$ 95	**744 $\pm$ 52**	620 $\pm$ 132	685 $\pm$ 72
Walker, stand	946 $\pm$ 8	946 $\pm$ 17	947 $\pm$ 9	948 $\pm$ 12	947 $\pm$ 5	**957 $\pm$ 11**	920 $\pm$ 40	944 $\pm$ 11
Cartpole, swingup	838 $\pm$ 38	824 $\pm$ 23	858 $\pm$ 5	**870 $\pm$ 8**	869 $\pm$ 10	845 $\pm$ 23	867 $\pm$ 9	825 $\pm$ 34
Cartpole, balance	940 $\pm$ 24	950 $\pm$ 57	964 $\pm$ 41	989 $\pm$ 6	**990 $\pm$ 9**	979 $\pm$ 23	**990 $\pm$ 9**	983 $\pm$ 4
Ball in cup, catch	659 $\pm$ 387	556 $\pm$ 233	750 $\pm$ 277	793 $\pm$ 149	648 $\pm$ 364	669 $\pm$ 285	703 $\pm$ 299	**884 $\pm$ 100**
Finger, spin	732 $\pm$ 201	722 $\pm$ 103	730 $\pm$ 144	773 $\pm$ 148	788 $\pm$ 144	848 $\pm$ 15	636 $\pm$ 117	**862 $\pm$ 2**
Finger, turn_easy	201 $\pm$ 35	226 $\pm$ 49	303 $\pm$ 125	349 $\pm$ 104	367 $\pm$ 38	404 $\pm$ 61	296 $\pm$ 51	**443 $\pm$ 77**
Cheetah, run	225 $\pm$ 29	245 $\pm$ 35	219 $\pm$ 31	**296 $\pm$ 54**	287 $\pm$ 91	246 $\pm$ 11	229 $\pm$ 39	273 $\pm$ 22
Average	645	629	674	709	702	712	658	**737**

TABLE III: Cumulative reward obtained in the 'color_easy' test environments. The highest scores are highlighted in bold and the results of the MSSA variations are marked in red if they are better than those of all the baselines.

Env	SAC	RAD	PAD	MSSA (ROT+ID+CURL)	MSSA (ID+CURL)	MSSA (ROT+CURL)	MSSA (ROT+ID)	MSSA (w/o Gray Aug)
Walker, walk	529 $\pm$ 6	515 $\pm$ 102	555 $\pm$ 95	565 $\pm$ 120	651 $\pm$ 139	**652 $\pm$ 65**	580 $\pm$ 142	512 $\pm$ 30
Walker, stand	894 $\pm$ 21	914 $\pm$ 30	906 $\pm$ 27	935 $\pm$ 20	**939 $\pm$ 12**	938 $\pm$ 11	874 $\pm$ 101	889 $\pm$ 34
Cartpole, swingup	733 $\pm$ 48	820 $\pm$ 21	761 $\pm$ 30	866 $\pm$ 8	**867 $\pm$ 8**	841 $\pm$ 21	860 $\pm$ 7	743 $\pm$ 39
Cartpole, balance	922 $\pm$ 31	945 $\pm$ 64	945 $\pm$ 30	981 $\pm$ 3	987 $\pm$ 7	967 $\pm$ 18	**988 $\pm$ 8**	959 $\pm$ 5
Ball in cup, catch	602 $\pm$ 256	608 $\pm$ 175	645 $\pm$ 236	732 $\pm$ 184	667 $\pm$ 280	540 $\pm$ 321	632 $\pm$ 357	**779 $\pm$ 84**
Finger, spin	711 $\pm$ 188	647 $\pm$ 202	709 $\pm$ 135	755 $\pm$ 144	770 $\pm$ 156	816 $\pm$ 20	616 $\pm$ 120	**853 $\pm$ 17**
Finger, turn_easy	219 $\pm$ 38	238 $\pm$ 34	305 $\pm$ 108	328 $\pm$ 103	**361 $\pm$ 9**	354 $\pm$ 69	255 $\pm$ 78	337 $\pm$ 66
Cheetah, run	199 $\pm$ 33	235 $\pm$ 44	180 $\pm$ 33	**261 $\pm$ 25**	258 $\pm$ 47	239 $\pm$ 10	212 $\pm$ 30	232 $\pm$ 58
Average	601	615	626	678	**688**	668	627	663

multiple self-supervised tasks can lead to better generalization compared to one. In addition, it is observed that the original MSSA trained with all three auxiliary tasks is the most robust among all the variations as it consistently outperforms the baselines. Moreover, among the self-supervised learning tasks, CURL appears to be more essential since MSSA (ROT+ID) generally results in worse performance, while the rotation prediction task seems to be less critical as MSSA (ID+CURL) does not demonstrate very obvious performance degradation. Meanwhile, the inverse dynamics prediction task demonstrates its importance in tasks with the 'goal-directed' nature, such as the 'catch', 'balance' and 'swingup' tasks. Besides, it is noticed that the grayscale data augmentation generalizes well to the randomized color distribution, however, it generalizes comparably worse to videos since MSSA (w/o Gray Aug) leads to better performance in the 'video' environments. Hence, better combinations of data augmentations need to be explored. So far, we only apply the random crop and grayscale augmentations for concept proof while it is important to find an optimal set of augmentations. Last but not least, since MSSA does not implement online learning during deployment, it has no adaptation delay and its inference time is similar to that of SAC and RAD. However, since the self-supervised learning networks can be trained without reward signals, MSSA can also implement online learning as PAD to achieve better performance in more challenging scenarios.

IV. Conclusion

In this paper, we propose a Multi-task Self-Supervised Adaptation (MSSA) method to address the generalization

TABLE IV: Cumulative reward obtained in the ‘color_hard’ environments. The highest scores are highlighted in bold and the results of the MSSA variations are marked in red if they are better than those of all the baselines.

Env	SAC	RAD	PAD	MSSA (ROT+ID+CURL)	MSSA (ID+CURL)	MSSA (ROT+CURL)	MSSA (ROT+ID)	MSSA (w/o Gray Aug)
Walker, walk	369 ± 10	409 ± 63	408 ± 80	415 ± 93	$\mathbf{519 \pm 133}$	498 ± 50	486 ± 165	356 ± 22
Walker, stand	716 ± 24	827 ± 66	784 ± 25	849 ± 6	850 ± 58	$\mathbf{864 \pm 13}$	765 ± 141	737 ± 53
Cartpole, swingup	536 ± 122	757 ± 11	575 ± 13	801 ± 10	$\mathbf{814 \pm 30}$	775 ± 27	785 ± 13	520 ± 93
Cartpole, balance	819 ± 33	889 ± 91	815 ± 76	907 ± 22	922 ± 34	880 ± 85	$\mathbf{941 \pm 14}$	798 ± 76
Ball in cup, catch	443 ± 205	521 ± 55	488 ± 151	527 ± 288	563 ± 292	465 ± 312	$\mathbf{667 \pm 279}$	500 ± 112
Finger, spin	592 ± 180	563 ± 243	625 ± 121	658 ± 119	731 ± 56	658 ± 71	530 ± 131	$\mathbf{742 \pm 25}$
Finger, turn_easy	187 ± 30	211 ± 39	269 ± 96	305 ± 27	323 ± 9	336 ± 29	284 ± 17	$\mathbf{343 \pm 52}$
Cheetah, run	157 ± 13	178 ± 65	139 ± 57	198 ± 30	$\mathbf{213 \pm 13}$	201 ± 27	177 ± 25	165 ± 78
Average	477	544	513	583	**617**	585	579	520

TABLE V: Cumulative reward obtained in the ‘video’ environment. The highest scores are highlighted in bold and the results of the MSSA variations are marked in red if they are better than those of all the baselines.

Env	SAC	RAD	PAD	MSSA (ROT+ID+CURL)	MSSA (ID+CURL)	MSSA (ROT+CURL)	MSSA (ROT+ID)	MSSA (w/o Gray Aug)
Walker, walk	491 ± 24	435 ± 72	520 ± 65	446 ± 88	511 ± 90	478 ± 60	464 ± 52	$\mathbf{538 \pm 38}$
Walker, stand	851 ± 66	787 ± 16	888 ± 54	765 ± 25	725 ± 21	724 ± 35	704 ± 46	$\mathbf{917 \pm 21}$
Cartpole, swingup	434 ± 113	441 ± 69	458 ± 82	523 ± 62	$\mathbf{527 \pm 19}$	485 ± 51	506 ± 68	474 ± 45
Cartpole, balance	$\mathbf{747 \pm 77}$	585 ± 93	660 ± 123	615 ± 69	599 ± 18	572 ± 25	599 ± 21	638 ± 60
Ball in cup, catch	350 ± 166	428 ± 129	261 ± 113	477 ± 89	428 ± 99	390 ± 81	$\mathbf{547 \pm 201}$	469 ± 64
Finger, spin	423 ± 102	303 ± 92	469 ± 120	379 ± 42	384 ± 81	390 ± 38	267 ± 93	$\mathbf{476 \pm 38}$
Finger, turn_easy	17 ± 26	3 ± 5	132 ± 144	211 ± 120	239 ± 91	227 ± 106	210 ± 179	$\mathbf{299 \pm 112}$
Cheetah, run	170 ± 6	150 ± 53	143 ± 43	145 ± 11	145 ± 20	129 ± 30	129 ± 21	$\mathbf{179 \pm 52}$
Average	435	392	441	445	445	424	428	**499**

challenge for RL agents. Our MSSA jointly trains the RL policy with multiple self-supervised objectives. It is worth mentioning that MSSA is a general paradigm that can be implemented on top of any RL algorithm without incurring longer inference time. Experimental results have demonstrated the remarkable superiority of MSSA.

Acknowledgment

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its Career Development Award (Grant No. C210112046).

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] K. Wu, M. Abolfazli Esfahani, S. Yuan, and H. Wang, “Learn to steer through deep reinforcement learning,” *Sensors*, vol. 18, no. 11, p. 3650, 2018.

[3] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, “Scalable deep reinforcement learning for vision-based robotic manipulation,” in *Conference on Robot Learning*, 2018, pp. 651–673.

[4] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, “Bnd*-ddqn: Learn to steer autonomously through deep reinforcement learning,” *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

[5] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, “Learn to navigate autonomously through deep reinforcement learning,” *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 5342–5352, 2021.

[6] K. Wu, M. Wu, J. Yang, Z. Chen, Z. Li, and X. Li, “Deep reinforcement learning boosted partial domain adaptation.”

[7] S. Gamrian and Y. Goldberg, “Transfer learning for related reinforcement learning tasks via image-to-image translation,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 2063–2072.

[8] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.

[9] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, “Using simulation and domain adaptation to improve efficiency of deep robotic grasping,” in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250.

[10] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, “Reinforcement learning with augmented data,” *arXiv preprint arXiv:2004.14990*, 2020.

[11] I. Kostrikov, D. Yarats, and R. Fergus, “Image augmentation is all you need: Regularizing deep reinforcement learning from pixels,” *arXiv preprint arXiv:2004.13649*, 2020.

[12] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus, “Automatic data augmentation for generalization in deep reinforcement learning,” *arXiv preprint arXiv:2006.12862*, 2020.

[13] N. Hansen, Y. Sun, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang, “Self-supervised policy adaptation during deployment,” *arXiv preprint arXiv:2007.04309*, 2020.

[14] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” *arXiv preprint arXiv:1803.07728*, 2018.

[15] A. Srinivas, M. Laskin, and P. Abbeel, “Curl: Contrastive unsupervised representations for reinforcement learning,” *arXiv preprint arXiv:2004.04136*, 2020.

[16] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq *et al.*, “Deepmind control suite,” *arXiv preprint arXiv:1801.00690*, 2018.

[17] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.

[18] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” *arXiv preprint arXiv:2002.05709*, 2020.