Visual Reinforcement Learning with Residual Action

Zhenxian Liu[1], **Peixi Peng**[2,3*], **Yonghong Tian**[1,2,3*]

[1]National Engineering Research Center of Visual Technology, School of Computer Science, Peking University, China
[2]School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University, China
[3]Peng Cheng Laboratory, China
zhenxianliu@stu.pku.edu.cn, {pxpeng, yhtian}@pku.edu.cn

Abstract

Learning control policy from continuous action space by visual observations is a fundamental and challenging task in reinforcement learning (RL). An essential problem is how to accurately map the high-dimensional images to the optimal actions by the policy network. Traditional decision-making modules output actions solely based on the current observation, while the distributions of optimal actions are dependent on specific tasks and cannot be known priorly, which increases the learning difficulty. To make the learning easier, we analyze the action characteristics in several control tasks, and propose Reinforcement Learning with **Res**idual **Act**ion (*ResAct*) to explicitly model the adjustments of actions based on the differences between adjacent observations, rather than learning actions directly from observations. The method just redefines the output of the policy network, and doesn't introduce any prior assumption to constrain or simplify the vanilla control problem. Extensive experiments on DeepMind Control Suite and CARLA demonstrate that the method could improve different RL baselines significantly, and achieve state-of-the-art performance.

1 Introduction

Learning continuous control from visual observations is a critical problem in reinforcement learning (RL) (Sutton and Barto 2018). The successful integration of RL algorithms (Schulman et al. 2017; Hessel et al. 2018; Haarnoja et al. 2018) with convolutional neural networks (CNNs) (LeCun et al. 1998) has demonstrated effectiveness in learning control policies from images, leading to remarkable achievements in video game playing (Mnih et al. 2013, 2015; Berner et al. 2019), classical board games (Silver et al. 2016, 2017), real-world robot grasping (Zhu et al. 2020; Kalashnikov et al. 2018), and autonomous navigation (Dosovitskiy et al. 2017).

Compared to learning from proprioceptive states, images reveal the intricate details of a complex and unstructured world (Zhang et al. 2020), thus still exhibiting significant shortcomings. Hence, a considerable body of research has focused on bridging the gap between vision-based and state-based (using proprioceptive states as inputs) RL regarding sample efficiency and asymptotic performance. An essential problem is how to accurately map the high-dimensional images to the optimal actions by policy network. To achieve this goal, existing works focus on representation learning (Lee et al. 2020; Laskin, Srinivas, and Abbeel 2020; Zhu et al. 2022; Choi et al. 2023), data augmentation (Yarats, Kostrikov, and Fergus 2020; Laskin et al. 2020; Hansen, Su, and Wang 2021), dynamic modeling of the environment (Gelada et al. 2019; Yu et al. 2021; Hafner et al. 2019b,a), and improving CNN optimization (Zhai et al. 2023). Besides the learning constraints and network inputs, the target outputs also influence the fitting ability of the network. Traditional decision-making modules output actions solely based on the current observation, while the distributions of optimal actions are dependent on specific tasks and cannot be known priorly, which increases the learning difficulty.

Hence, it is necessary to analyze the underlying action characteristics in realistic control tasks:

(1) Smooth policies are beneficial for a wide range of RL tasks as stated in (Mysore et al. 2021; Shen et al. 2020). We experimentally investigate different vision-based RL methods—including pixel SAC, RAD (Laskin et al. 2020), DeepMDP (Gelada et al. 2019), and the combination of DeepMDP and RAD (hereinafter referred to as DeepRAD)—we quantify the smoothness of decision-making by calculating the mean Euclidean distance between adjacent actions across three tasks of CARLA (Dosovitskiy et al. 2017) and OpenAI DMControl (Tassa et al. 2018) on 10 evaluation trajectories after training. This metric is referred to as the **M**ean **A**ction **D**istance (*MAD*). To display the results of different tasks on a single plot, we normalize performance and MAD for each task. Fig. 1(a) indicates that methods with lower MAD generally perform better, suggesting that smooth action adjustments over adjacent time steps are beneficial.

(2) Based on the action smoothness, most of the residual actions (i.e., the difference value between adjacent actions) will be very small or even close to zero, and only a small proportion of residual actions has a large value. To validate this point, we take the policy learned by state-based RL as an approximation of the optimal policy and count the distribution of actions and residual actions. As shown in Fig. 1(b), unlike the scattered distribution of actions across the value range, **residual actions generally follow a normal distribution centered around zero, providing a simpler**

*Corresponding author.

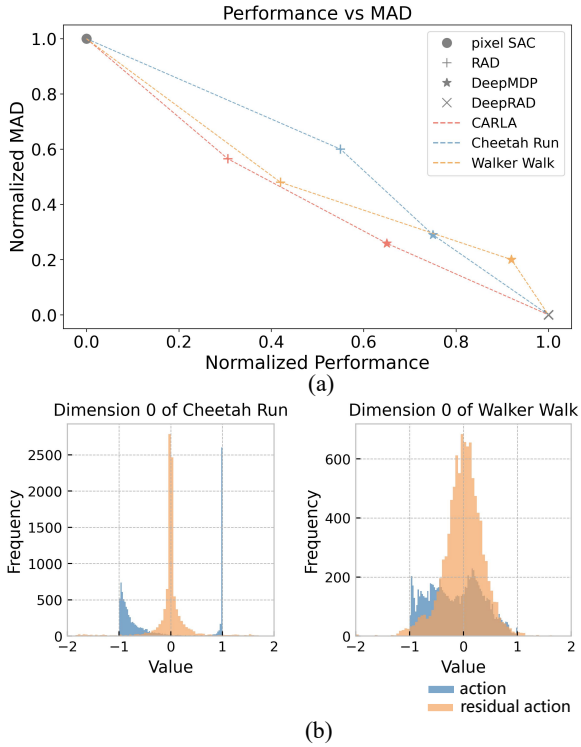

Figure 1: (a) Statistical analysis of MAD and performance. The points in the bottom-right corner and the top-left corner are shared by three tasks (indicating that the best-performing algorithm always has the lowest MAD, and vice versa). (b) Distribution of actions and residual actions of state-based SAC on Cheetah Run and Walker Walk. For brevity, we only show the distribution on the first dimension.

distribution that is easier to explore. Several visual comparisons of learning actions and residual actions are shown in Appendix E.2.

(3) It is hard to explicitly and directly build a connection between complex visual observation and action, but it is clear that the large residual action is always caused by the obvious changes in observations. Fig. 2 is a visualization example of learned latent representations of different learning paradigms. **This indicates that the mapping from the representation space of observation difference to residual action space is simpler than mapping the individual observations to actions.** More details of Fig. 2 are shown in Appendix E.1.

Building on these insights, we propose Reinforcement Learning with **Res**idual **Act**ion (*ResAct*) for vision-based RL. Our analysis of MAD and residual action distribution uncovers the potential correlation between the smoothness of action sequences and algorithm performance. Instead of modeling complete action based solely on current observation, our policy network incorporates previous action into the decision-making process and predicts a residual action. Such modeling of residual actions encourages the agent to explore smooth policy in priority. In addition, the optimal action distributions vary from different tasks and it is hard

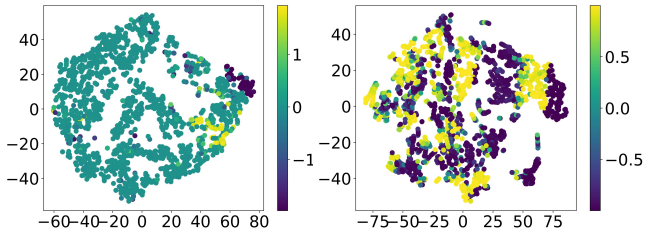

Figure 2: The t-SNE diagrams of latent spaces learned using our method (left) and baseline DeepRAD (right) after training, colored according to residual action values (for ours) or action values(for DeepRAD).

to know them priorly in vanilla RL, while the residual actions' distributions could be assumed as a normal distribution centered around zero in most cases, which is useful for network initialization. Considering the unusual worst case where the agent changes actions dramatically at each step, our assumption will fail and the effectiveness is similar to the vanilla RL where the prior action distributions are unknown. We explore the correlation between Residual Action Learning and human behavior patterns in Appendix H.1.

While **R**esidual **A**ction **L**earning (*RAL*) is expected to enhance the overall performance by encouraging incremental action refinements, as we are now modeling residual action instead of action, we posit that a more intuitive and applicable representation of visual observations is needed. To this end, we introduce **O**bservation **D**ifference **L**earning (*ODL*). ODL captures dynamic changes between adjacent observations by separately inputting them into two structurally identical but independently parameterized CNNs, and then subtracting the resulting feature maps to extract differences in observations. These subtracted feature maps are then compressed into fixed-size feature vectors, which are subsequently used for learning residual actions. Note that (Shang et al. 2021) also utilizes differencing of feature maps. The main difference is that they calculate the difference between single-frame images to extract temporal information, while ODL extracts observation differences with frame-stacked observations to better fit the residual action.

Note that there exist several methods that aim to learn smooth policy by regularization (Shen et al. 2020; Mysore et al. 2021) or introducing additional rewards (Mahmood et al. 2018; Molchanov et al. 2019; Koch III 2019). They force the learned policy to be smooth which may fail when the agent needs to change actions dramatically in abrupt cases. Compared with them, our method adjusts the learning paradigm rather than constrain the policy distribution. That is, we do not introduce any prior assumption to constrain or simplify the vanilla control problem and do not influence the theoretical optimality of RL. This is the main difference compared with other methods based on the smooth assumption. Overall, the contributions of this paper are summarized as follows:

- We introduce a novel **R**esidual **A**ction **L**earning (*RAL*) which encourages agents to make incremental action adjustments rather than learn the action directly, thereby

significantly decreasing the learning difficulty.

- We propose **O**bservation **D**ifference **L**earning (*ODL*) to explicitly model the differences between adjacent observations, providing a more intuitive and compact representation for Residual Action Learning.
- Extensive experiments on OpenAI DeepMind Control Suite (DMControl) (Tassa et al. 2018) and CARLA (Dosovitskiy et al. 2017) have demonstrated that ResAct significantly surpasses previous state-of-the-art. Codes and appendix have also been made public[1].

2 Related Work

2.1 Vision-based Reinforcement Learning

Enabling agents to learn control policies from visual inputs is a research area of significant interest. Simply combining RL algorithms such as Soft Actor-Critic (SAC) (Haarnoja et al. 2018), Proximal Policy Optimization (PPO) (Schulman et al. 2017), and Rainbow (Hessel et al. 2018) with CNNs has demonstrated potential. However, compared to state-based RL, vision-based RL still faces substantial gaps in performance. Efforts to bridge this gap generally focus on three strategies: (i) introducing auxiliary losses or learning tasks to enhance representation learning (Hansen et al. 2020; Castro 2020; Li et al. 2022; Stooke et al. 2021; Laskin, Srinivas, and Abbeel 2020), (ii) utilizing data augmentation techniques to improve generalization over visual inputs (Laskin et al. 2020; Yarats, Kostrikov, and Fergus 2020; Hansen, Su, and Wang 2021), and (iii) modeling environmental dynamics for a better understanding of state transitions (Gelada et al. 2019; Yu et al. 2021; Hafner et al. 2019a,b).

2.2 Smooth Policy in Reinforcement Learning

Smooth policy learning has been explored in the RL community but has not received widespread attention. By introducing regularization terms in learning objectives of the policy and value function, (Shen et al. 2020) guides the learning of smooth policies and enhances the agent's robustness to noise. (Mysore et al. 2021) introduces spatial and temporal smoothness regularization to policy network, aiming to address control signal oscillation in real-world RL scenarios. (Mahmood et al. 2018; Molchanov et al. 2019; Koch III 2019) engineer reward functions to encourage the generation of smooth control signals. These approaches indirectly encourage smooth control and introduce additional computational overhead or require domain knowledge, while we use smooth as motivation and propose ResAct to reduce the learning difficulty. We do not introduce any prior assumption to constrain or simplify the control task.

2.3 Residual Learning in RL

Deep residual networks (He et al. 2016), which incorporate additive residual blocks, have set new benchmarks in a variety of computer vision tasks by facilitating the training of extremely deep neural networks. Learning a residual policy that refines a conventional feedback control policy has been explored in RL. (Johannink et al. 2019) and (Zeng et al. 2020) combine the residual policy learned by RL with the base control policy predicted by the physics simulator to create an enhanced final control policy, which is particularly effective for complex manipulation tasks that involve intricate interactions like friction and contact with unstable objects. Similarly, RL has also been used to learn residual policies that are tailored to adjust control policies derived from demonstration data (Alakuijala et al. 2021). In essence, a base policy is first acquired through conventional feedback control or derived from demonstration data in previous works. Then, a residual policy relative to the base policy is learned. Two policies should be used together, while ours refers specifically to temporal residual actions.

[1]https://github.com/LiuZhenxian123/ResAct

3 Preliminary

Vision-based reinforcement learning (RL) can be formulated as a **P**artially **O**bservable **M**arkov **D**ecision **P**rocess (**POMDP**) (Kaelbling, Littman, and Cassandra 1998), represented by the tuple $(\mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. $\mathcal{O}$ is the observation space consisting of image frames o_t at different time step t. $\mathcal{A}$ is the action space, including all possible actions a_t that can be taken by the agent. $\mathcal{P}$, the state transition distribution $\mathcal{P}(o_{t+1}|o_t, a_t)$, defines how actions influence observation transitions. $\mathcal{R}(o_t, a_t)$ is the reward function, offering feedback to the agent's action on corresponding observation. $\gamma \in [0, 1)$, the discount factor, prioritizes immediate rewards over future rewards. The agent's goal is to maximize expected cumulative rewards, expressed by $R = \sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the reward at time t. During training, interactions are stored in a replay buffer $\mathcal{B}$.

Soft Actor-Critic The Soft Actor-Critic (SAC) (Haarnoja et al. 2018) algorithm is a leading off-policy method to solve continuous control problems. SAC optimizes a policy $\pi_\theta(a|o)$ and a critic $Q_\phi(o, a)$ with the objective of maximizing the expected sum of rewards and the entropy of the policy, formalized as $\sum_t \mathbb{E}_{o_t, a_t \sim \pi_\theta} [r_t + \alpha \mathcal{H}(\pi_\theta(a_t|o_t))]$. During the interaction with the environment, the action value Q is estimated by minimizing the soft Bellman error:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{\tau \sim \mathcal{B}} \left[\left(Q_\phi(o_t, a_t) - (r_t + \gamma V(o_{t+1})) \right)^2 \right] \quad (1)$$

The target value for the next observation is approximated by sampling an action from the current policy:

$$V(o_{t+1}) = \mathbb{E}_{a' \sim \pi} \left[Q_{\bar{\phi}}(o_{t+1}, a') - \alpha \log \pi_\theta(a'|o_{t+1}) \right] \quad (2)$$

where the weights of $Q_{\bar{\phi}}$ is updated as exponentially moving average of Q_ϕ . The policy is updated by the equation below:

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_{a \sim \pi} \left[Q_\phi(o_t, a) - \alpha \log \pi_\theta(a|o_t) \right] \quad (3)$$

The policy action is sampled using the reparameterization trick, which is also employed in the updating of the temperature parameter α to approximate the target entropy.

4 Methodology

4.1 Residual Action Learning

To learn policy from continuous action space, nearly all reinforcement learning (RL) algorithms adopt a decision-making mechanism that directly outputs actions solely based

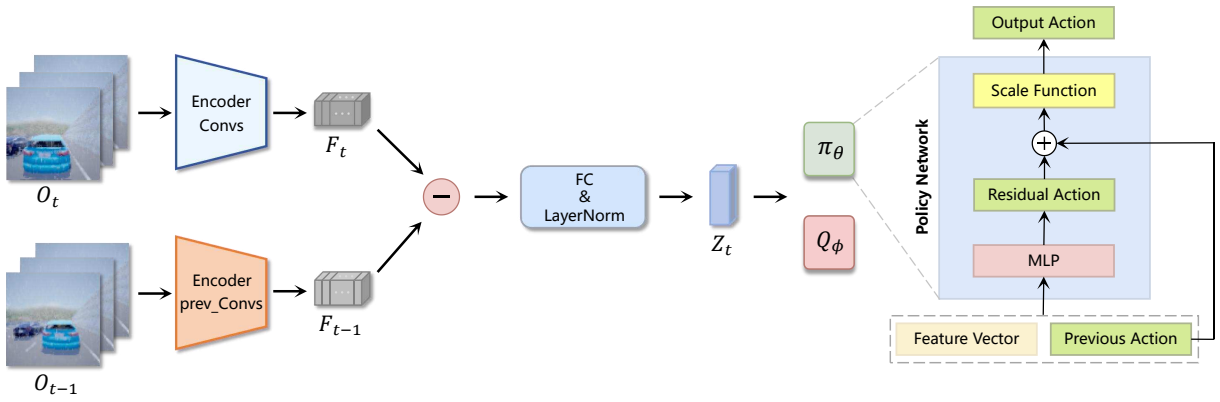

Figure 3: Framework of ResAct. It compromises two parts: Observation Difference Learning (*ODL, on the left*) employs structurally identical but independently parameterized CNNs to encode adjacent observations, further compressing the subtracted feature map into a fixed-length feature vector. Residual Action Learning (*RAL, on the right*) enables the policy network to learn a residual action relative to the previous action, then the sum of both is scaled to get the final action to execute.

on current observations. This mechanism has remained unchanged since its inception (Sutton and Barto 2018). Through statistical analysis in Sec. 1, we argue that learning residual action is much easier than the action itself. Following this insight, we seek a general alternative to explicitly guide the policy network to output residual action, which could be coupled to most base RL algorithms with minimal modification. Therefore, we propose RAL to explicitly model the action adjustments. Instead of letting the policy network directly output action given the current observation, we take the current observation together with the previous action as inputs to the policy network and output a residual action. The residual action is then added to the previous action to form the final action, as shown in Fig. 3.

4.2 Observation Difference Learning

The principle driving us to propose ODL is quite straightforward: We consider the root factor determining the adjustment of actions is the differences in observations. Therefore, by modeling the differences in adjacent observations through ODL, we aim to provide a more compact and suitable visual representation for RAL. By inputting the current observation o_t and the previous observation o_{t-1} into two structurally identical but independently parameterized CNNs, we obtain two feature maps of dimension $c \times h \times w$. Taking the difference between these two feature maps serves as the input to a fully connected layer (FC) with Layer Normalization (LN), further compressing it into a fixed-length feature vector z_t which suppresses redundant information between adjacent observations while preserving information of dynamic changes.

Regarding the parameters of the CNN used for processing the previous observation, we also experimented with allowing it to share parameters or momentum update (He et al. 2020) with the CNN that processes the current observation, but the results were not as satisfactory as using two CNNs with independent parameters. Although experiments in Fig. 4 indicate that using RAL alone can already bring significant gains to the baseline, ODL alleviates the consistent information between adjacent observations while retaining the information of dynamic changes. With only minimal modifications to the original encoder, ODL provides a more compact and effective feature vector for RAL, further enhancing the overall performance.

Algorithm 1: Pseudocode for Inference Procedure

Black: unmodified Soft Actor-Critic.
Orange: Observation Difference Learning.
Blue: Residual Action Learning.

Given π_θ, f_{convs}, $f_{\text{prev\_convs}}$;
for *each timestep* t **do**
  // Observation Difference Learning (ODL)
  $F_{\text{obs}} = f_{\text{convs}}(o_t)$, $F_{\text{prev\_obs}} = f_{\text{prev\_convs}}(o_{t-1})$;
  $F_{\text{obs\_difference}} = F_{\text{obs}} - F_{\text{prev\_obs}}$;
  $z_t = \text{LayerNorm}(f_{\text{FC}}(F_{\text{obs\_difference}}))$;
  // Residual Action Learning (RAL)
  $\delta_{a_t} \sim \pi_\theta(\cdot|z_t, a_{t-1})$;
  $a_t = \tanh(a_{t-1} + \delta_{a_t})$;
  $o_{t+1} \sim p(o_{t+1}|o_t, a_t)$;
end for

4.3 Method Analysis

Exploration Reinforcement learning algorithms typically make dozens to hundreds of decisions per second. The high frequency of decision-making means that the changes in adjacent observations are often small. The high similarity in adjacent observations calls for slight variations in actions. In most cases, the action taken in the previous time step can be a reasonable choice for the current time step. A small residual action is expected to be sampled, which can provide appropriate explorations around the previous action. In some rare cases where observations change drastically, more aggressive exploration is often required. Since we have not introduced any constraints into the overall RL framework, a large residual action still can be taken. More specifically, taking SAC as an example, a Gaussian distribution of resid-

100K STEP SCORES	pixel SAC	+ResAct	DeepMDP	+ResAct	RAD	+ResAct	DeepRAD	+ResAct
Cartpole, Swingup	237 ± 49	446 ± 39	389 ± 44	482 ± 46	694 ± 28	785 ± 72	703 ± 36	819 ± 44
Reacher, Easy	239 ± 183	465 ± 143	471 ± 173	562 ± 129	734 ± 87	871 ± 62	792 ± 77	917 ± 59
Cheetah, Run	118 ± 13	284 ± 29	306 ± 25	382 ± 42	364 ± 38	477 ± 45	453 ± 39	503 ± 42
Walker, Walk	95 ± 19	358 ± 31	384 ± 197	541 ± 165	552 ± 87	692 ± 101	582 ± 91	772 ± 65
Finger, Spin	230 ± 194	486 ± 150	509 ± 72	687 ± 64	813 ± 65	911 ± 47	832 ± 101	974 ± 42
Ball in cup, Catch	85 ± 130	243 ± 61	704 ± 24	728 ± 33	825 ± 49	871 ± 67	809 ± 45	948 ± 44
Average	167.3	380.3	460.5	563.6	663.6	767.8	695.1	822.1
500K STEP SCORES								
Cartpole, Swingup	330 ± 73	658 ± 87	817 ± 15	802 ± 23	861 ± 9	863 ± 29	870 ± 6	870 ± 12
Reacher, Easy	307 ± 65	681 ± 66	792 ± 83	873 ± 98	917 ± 47	951 ± 37	942 ± 45	974 ± 16
Cheetah, Run	85 ± 51	421 ± 53	613 ± 32	664 ± 42	669 ± 42	728 ± 24	721 ± 58	750 ± 8
Walker, Walk	71 ± 52	563 ± 46	594 ± 172	792 ± 133	907 ± 43	923 ± 34	926 ± 28	953 ± 21
Finger, Spin	346 ± 95	655 ± 130	828 ± 63	912 ± 117	922 ± 48	951 ± 30	932 ± 92	979 ± 4
Ball in cup, Catch	162 ± 122	510 ± 122	944 ± 16	951 ± 13	959 ± 15	960 ± 14	954 ± 11	967 ± 4
Average	216.8	581.3	764.6	832.3	872.5	896.0	890.8	915.5

Table 1: Implement ResAct on top of four different baselines. *+ResAct* represents the preceding method combined with ResAct. ResAct is capable of bringing improvements to all baselines.

ual actions centered far from the previous action can be learned to make more aggressive explorations in the action space, which does not increase the learning difficulty compared to vanilla RL methods.

Learning During model initialization, weights are typically set to very small values, so the predicted residual actions are almost around zero. As mentioned in Section 1, the ideal residual actions are mostly distributed around zero. Therefore, the distance between the model's prediction and the target is not far in the beginning, and it is expected to receive smoother gradient signals during training. Visualizations in Appendix E.2 indicate that the residual actions learned by ResAct better approximate the ideal distribution.

4.4 Implementation Details

We select DeepRAD as our baseline, which combines RAD's (Laskin et al. 2020) data augmentation with DeepMDP's (Gelada et al. 2019) transition loss and reward loss. Building upon DeepRAD, the introduction of ResAct requires only minor modifications and does not necessitate any additional hyperparameters. The pseudocode illustrates inference with ResAct in Algorithm 1.

5 Experiments

In this section, we first demonstrate significant improvements in sample efficiency and asymptotic performance brought by ResAct to baseline DeepRAD on two widely used benchmarks in RL, namely DMControl (Tassa et al. 2018)) and CARLA (Dosovitskiy et al. 2017). Subsequently, we conduct multiple ablation experiments to dive deeper into the design choices of ResAct.

ResAct requires only minor modifications to the baseline and does not introduce any additional hyperparameters. For a fair comparison, we have adopted the settings for model structures and hyperparameters used by a range of approaches (Laskin, Srinivas, and Abbeel 2020; Laskin et al. 2020; Yarats, Kostrikov, and Fergus 2020; Zhang et al. 2020). For the DMControl benchmark, we begin by presenting experimental results on 6 heavily benchmarked DMControl common tasks and further highlight the significant improvements of ResAct on DMControl 5 hard tasks demonstrated by Flare (Shang et al. 2021). We choose the highway driving task of CARLA to evaluate ResAct on scenarios with more complex observations. We largely follow the settings in DBC (Zhang et al. 2020), where the agent's goal is to drive as far along the figure-8 highway of CARLA's Town04 as possible in 1000 time steps without collision. Experiments are conducted using 5 seeds and the mean and standard deviation of rewards are reported. For detailed settings of the experiments, please refer to Appendix G.

5.1 Main Results

DMControl 6 common tasks: The 6 tasks presented in Table 1 and 2 are heavily benchmarked within DMControl. We first introduce ResAct on several typical different baselines including Pixel SAC (Haarnoja et al. 2018), DeepMDP (Gelada et al. 2019) and RAD (Laskin et al. 2020). Since DeepMDP learns the representation by predicting state transition and future rewards, and RAD is based on data augmentation. Hence, we combine these two methods as a stronger baseline named DeepRAD. As shown in Table 1, ResAct consistently enhances the performance of various baselines, demonstrating its excellent adaptability.

Then we meticulously select a range of methods for comparison with ResAct such as CURL (Laskin, Srinivas, and Abbeel 2020), MLR (Yu et al. 2022), PSRL (Choi et al. 2023), SVEA (Hansen, Su, and Wang 2021), PlayVirtual (Yu et al. 2021), MaDi (Grooten et al. 2023) and TACO (Zheng et al. 2024). As shown in Table 2, our method achieves state-of-the-art performance in both sample efficiency(**5 out of 6**) and asymptotic performance(**4 out of 6**) for the majority of tasks, while being competitive in the remaining tasks. Moreover, the standard deviation of the reward is also significantly lower, indicating ResAct also has advantages in convergence stability and ease of optimization.

100K STEP SCORES Reference	**CURL** ICML'20	**SVEA** NeurIPS'21	**PlayVirtual** NeurIPS'21	**MLR** NeurIPS'22	**PSRL** CVPR'23	**TACO** NeurIPS'23	**MaDi** AAMAS'24	**ResAct** This work
Cartpole, Swingup	582 ± 146	727 ± 86	816 ± 36	806 ± 48	$\mathbf{849 \pm 63}$	782 ± 51	704 ± 54	819 ± 44
Reacher, Easy	538 ± 233	811 ± 115	785 ± 142	866 ± 103	621 ± 202	821 ± 97	766 ± 101	$\mathbf{917 \pm 59}$
Cheetah, Run	299 ± 48	375 ± 54	474 ± 50	482 ± 38	398 ± 71	402 ± 62	432 ± 44	$\mathbf{503 \pm 42}$
Walker, Walk	403 ± 24	747 ± 65	460 ± 173	643 ± 114	595 ± 104	601 ± 103	574 ± 94	$\mathbf{772 \pm 65}$
Finger, Spin	767 ± 56	859 ± 77	915 ± 49	907 ± 58	882 ± 132	876 ± 67	810 ± 95	$\mathbf{974 \pm 42}$
Ball in cup, Catch	769 ± 43	915 ± 71	929 ± 31	933 ± 16	922 ± 60	902 ± 54	884 ± 36	$\mathbf{948 \pm 44}$
Average	559.7	739.0	729.8	772.8	711.1	730.7	695.0	**822.1**
500K STEP SCORES								
Cartpole, Swingup	841 ± 45	865 ± 10	865 ± 11	872 ± 5	$\mathbf{895 \pm 39}$	870 ± 21	849 ± 6	870 ± 12
Reacher, Easy	929 ± 44	944 ± 52	942 ± 66	957 ± 41	932 ± 41	944 ± 50	955 ± 31	$\mathbf{974 \pm 16}$
Cheetah, Run	518 ± 28	682 ± 65	719 ± 51	674 ± 37	686 ± 80	663 ± 30	732 ± 45	$\mathbf{750 \pm 8}$
Walker, Walk	902 ± 43	919 ± 24	928 ± 30	939 ± 10	930 ± 75	914 ± 87	912 ± 26	$\mathbf{953 \pm 21}$
Finger, Spin	926 ± 45	924 ± 93	963 ± 40	973 ± 31	961 ± 121	972 ± 89	951 ± 47	$\mathbf{979 \pm 4}$
Ball in cup, Catch	959 ± 27	960 ± 19	967 ± 5	964 ± 14	$\mathbf{988 \pm 54}$	960 ± 22	912 ± 62	967 ± 4
Average	845.8	882.3	897.3	896.5	894.1	887.1	885.1	**915.5**

Table 2: Performance comparison with SOTA methods on DMControl 6 common tasks at 100K and 500K environment steps. We apply ResAct on top of DeepRAD and highlight it in gray. The best results are highlighted in **bold**.

Task	**Flare**	**TACO**	**MaDi**	**DeepRAD**	**ResAct(500K)**	**Flare**	**TACO**	**MaDi**	**DeepRAD**	**ResAct(1M)**
Quadruped, Walk	296 ± 139	345 ± 89	277 ± 92	307 ± 142	$\mathbf{385 \pm 81}$	488 ± 221	665 ± 144	621 ± 172	586 ± 193	$\mathbf{690 \pm 128}$
Pendulum, Swingup	242 ± 152	485 ± 167	372 ± 101	308 ± 137	$\mathbf{618 \pm 380}$	809 ± 31	784 ± 42	751 ± 41	626 ± 220	$\mathbf{817 \pm 6}$
Hopper, Hop	90 ± 55	$\mathbf{112 \pm 42}$	80 ± 24	51 ± 19	99 ± 49	217 ± 59	221 ± 45	201 ± 43	212 ± 13	$\mathbf{233 \pm 32}$
Finger, Turn hard	282 ± 67	372 ± 174	311 ± 143	173 ± 195	$\mathbf{465 \pm 153}$	661 ± 315	672 ± 167	695 ± 133	310 ± 278	$\mathbf{857 \pm 80}$
Walker, Run	426 ± 33	355 ± 89	382 ± 87	375 ± 177	$\mathbf{467 \pm 27}$	556 ± 93	$\mathbf{582 \pm 63}$	562 ± 68	508 ± 125	554 ± 21
Average	267.2	333.8	284.4	242.8	**406.8**	546.2	584.8	566.0	448.4	**630.2**

Table 3: Performance on DMControl 5 hard tasks. Following the settings of Flare (Shang et al. 2021), we report the results at 500K and 1M environment steps.

DMControl 5 hard tasks: The DMControl 6 common tasks are considered solved with limited room for improvement since pixel-based methods are as efficient as state-based methods. Therefore, we continue to conduct experiments on five hard tasks selected by Flare (Shang et al. 2021), where there is still a large gap between pixel-based and state-based approaches, to stress the improvements brought by ResAct. We follow Flare's setting of using more training steps and compare our results with DeepRAD and Flare. In Table 3, the evaluations at 500K and 1M environment steps indicate that ResAct outperforms others on all tasks except for Walker Run, while ResAct significantly reduces the standard deviation (21 vs 63). It is also noteworthy that ResAct shows an overall reduction in the standard deviation of performance on all tasks at 1M environment steps, highlighting its advantage in convergence stability. Table 3 shows that ResAct proves advantageous for more challenging continuous control tasks that deal with partial observation, sparse rewards, or those require precise manipulation.

CARLA: The CARLA simulator features photo-realistic visual observations, which contain a huge variety of task-irrelevant information, making it an excellent choice for testing algorithm performance in more realistic scenarios. On the CARLA autonomous driving task, the advantages of ResAct are further validated. Table 4 clearly shows that ResAct has achieved substantial improvements in both episode reward and driving distance and has significantly reduced the average steer and brake, thus greatly enhancing the smoothness of the driving process.

5.2 Ablation Studies

Effectiveness of each component To separately explore the contributions of RAL and ODL, we incrementally introduce each independent module on top of DeepRAD in the CARLA autonomous driving task and plot Fig. 4(a). It can be seen that modeling residual actions using only RAL can significantly improve upon the baseline, which indicates the superiority of the decision-making mechanism based on residual actions. This allows for direct benefits from RAL without changing the encoder. The further introduction of ODL leads to an additional increase in performance. We hypothesize that in terms of learning residual actions, the representations of independent observations contain redundant information that the agent needs to learn to ignore, while ODL significantly suppresses this redundant information, thereby providing a more compact visual representation.

Dive into ODL In Section 4.2, we assume that ODL can alleviate redundant information between adjacent observations while retaining the information of dynamic changes, thus providing a more compact and suitable representation for RAL. To validate our hypothesis, we gradually reduce the feature dimension and compare it with baseline DeepRAD. As depicted in Fig. 4(b) and Appendix B.1, as the feature dimension is gradually compressed, our proposed

Method	Episode reward ↑	Distance (m) ↑	Crash intensity ↓	Average steer ↓	Average brake ↓
pixel SAC (Arxiv'18)	121 ± 26.1	74 ± 17.4	3930 ± 80.3	17.52% ± 0.021%	1.81% ± 0.013%
CURL (ICML'20)	134 ± 15.1	128 ± 32.5	3050 ± 100.3	16.60% ± 0.025%	2.94% ± 0.021%
RAD (NeurIPS'20)	142 ± 26.4	112 ± 33.6	2876 ± 94.9	16.80% ± 0.033%	2.12% ± 0.032%
DrQ (ICLR'21)	154 ± 21.5	95 ± 27.2	2419 ± 72.3	15.79% ± 0.018%	1.70% ± 0.039%
SVEA (NeurIPS'21)	161 ± 31.3	125 ± 18.9	2577± 71.1	13.22% ± 0.011%	1.43% ± 0.024%
Flare (NeurIPS'21)	132 ± 24.7	90 ± 14.6	2668 ± 95.7	11.48% ± 0.022%	1.52% ± 0.014%
ISO-Dream (NeurIPS'22)	117 ± 19.3	86 ± 23.6	3342 ± 97.7	18.82% ±0.013%	1.86% ± 0.027%
DeepMDP (ICML'19)	170 ± 36.1	132 ± 20.4	**2136 ± 69.3**	10.22% ± 0.015%	1.65% ± 0.007%
TACO (NeurIPS'23)	208 ± 23.4	197 ± 17.6	2997 ± 104.8	16.78% ±0.022%	1.58% ± 0.030%
MaDi (AAMAS'24)	177 ± 18.6	143 ± 28.9	2557 ± 86.3	14.46% ±0.035%	2.47% ± 0.024%
DeepRAD	198 ± 27.3	194 ± 21.7	2248 ± 79.8	9.32% ± 0.024%	1.03% ± 0.028%
ResAct	**283 ± 25.3**	**299 ± 24.6**	2744 ± 122.6	**7.07% ± 0.010%**	**0.30% ± 0.029%**

Table 4: Driving metrics at 100k training steps, with arrow directions indicating whether a larger or smaller value is better.

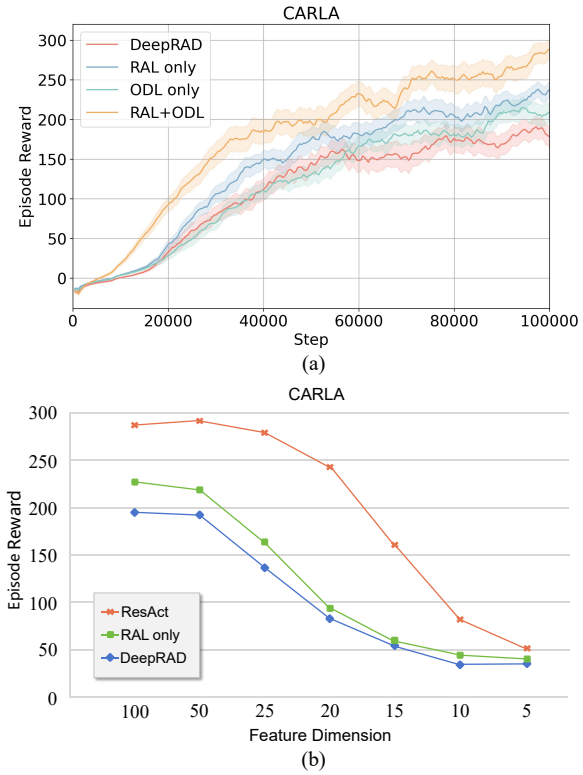

Figure 4: (a) Test performance comparison on CARLA. Gradually introducing RAL and ODL on top of DeepRAD brings continuous performance improvements. (b) Comparison of test performance on CARLA as the encoder feature dimension is progressively compressed. ODL is capable of providing a more compact representation.

method continues to maintain stable performance, whereas the performance of DeepRAD plummets. We also follow (Zagoruyko and Komodakis 2016) to compute ODL's attention map by performing mean pooling on the absolute values of the activations along channels and then applying a 2D spatial softmax. As shown in Fig. 5, In DMControl tasks, DeepRAD forces the agent to distribute its attention across the entire body of the robot, while ResAct allows the agent to selectively focus its attention on the edges of the torso and limbs where dynamic changes occur. In CARLA environment, the observations encompass a vast array of open-world elements, which significantly challenges the attention allocation for DeepRAD (noticing that it incorrectly focuses on some task-irrelevant elements). In contrast, ResAct is still capable of concentrating its attention on moving vehicles.

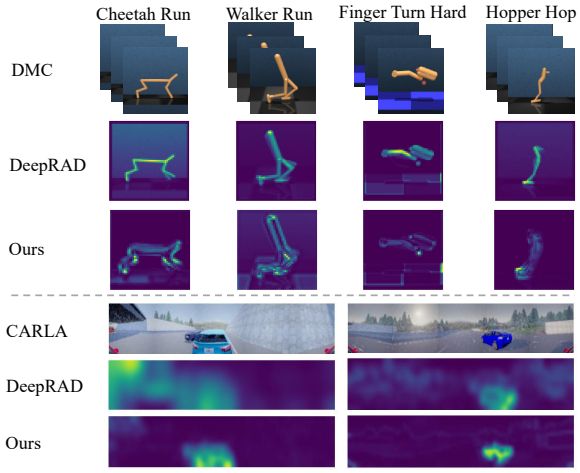

Figure 5: Spatial attention maps for different tasks. ODL enables the agent to narrow down the area of attention allocation, concentrating on regions with dynamic changes.

6 Conclusion

Motivated by the correlations between the smoothness of decision-making and algorithm performance, we propose Reinforcement Learning with **Res**idual **Act**ion (*ResAct*) for vision-based RL, which reduces the learning difficulty. Extensive experiments demonstrate significant improvements of ResAct in terms of sample efficiency and asymptotic performance. Our future work includes conducting more in-depth explorations of this new perspective on policy learning and applying ResAct in model-based frameworks.

Acknowledgments

The study was supported by the National Natural Science Foundation of China under contracts No. 62332002, No. 62425101, No. 62088102, No. 62422602, No. 62372010, No. 62027804, Key Laboratory Grants 241-HF-D05-01, and the major key project of the Peng Cheng Laboratory (PCL2021A13). Computing support was provided by Pengcheng Cloudbrain.

References

Alakuijala, M.; Dulac-Arnold, G.; Mairal, J.; Ponce, J.; and Schmid, C. 2021. Residual reinforcement learning from demonstrations. *arXiv preprint arXiv:2106.08050*.

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Castro, P. S. 2020. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 10069–10076.

Choi, H.; Lee, H.; Song, W.; Jeon, S.; Sohn, K.; and Min, D. 2023. Local-Guided Global: Paired Similarity Representation for Visual Reinforcement Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15072–15082.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*, 1–16. PMLR.

Gelada, C.; Kumar, S.; Buckman, J.; Nachum, O.; and Bellemare, M. G. 2019. Deepmdp: Learning continuous latent space models for representation learning. In *International conference on machine learning*, 2170–2179. PMLR.

Grooten, B.; Tomilin, T.; Vasan, G.; Taylor, M. E.; Mahmood, A. R.; Fang, M.; Pechenizkiy, M.; and Mocanu, D. C. 2023. MaDi: Learning to Mask Distractions for Generalization in Visual Deep Reinforcement Learning. *arXiv preprint arXiv:2312.15339*.

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

Hafner, D.; Lillicrap, T.; Ba, J.; and Norouzi, M. 2019a. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*.

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019b. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, 2555–2565. PMLR.

Hansen, N.; Jangir, R.; Sun, Y.; Alenyà, G.; Abbeel, P.; Efros, A. A.; Pinto, L.; and Wang, X. 2020. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*.

Hansen, N.; Su, H.; and Wang, X. 2021. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. *Advances in neural information processing systems*, 34: 3680–3693.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9729–9738.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Johannink, T.; Bahl, S.; Nair, A.; Luo, J.; Kumar, A.; Loskyll, M.; Ojea, J. A.; Solowjow, E.; and Levine, S. 2019. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, 6023–6029. IEEE.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2): 99–134.

Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. 2018. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, 651–673. PMLR.

Koch III, W. F. 2019. *Flight controller synthesis via deep reinforcement learning*. Ph.D. thesis, Boston University.

Laskin, M.; Lee, K.; Stooke, A.; Pinto, L.; Abbeel, P.; and Srinivas, A. 2020. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895.

Laskin, M.; Srinivas, A.; and Abbeel, P. 2020. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, 5639–5650. PMLR.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Lee, A. X.; Nagabandi, A.; Abbeel, P.; and Levine, S. 2020. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33: 741–752.

Li, X.; Shang, J.; Das, S.; and Ryoo, M. 2022. Does self-supervised learning really improve reinforcement learning from pixels? *Advances in Neural Information Processing Systems*, 35: 30865–30881.

Mahmood, A. R.; Korenkevych, D.; Vasan, G.; Ma, W.; and Bergstra, J. 2018. Benchmarking reinforcement learning algorithms on real-world robots. In *Conference on robot learning*, 561–591. PMLR.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Molchanov, A.; Chen, T.; Hönig, W.; Preiss, J. A.; Ayanian, N.; and Sukhatme, G. S. 2019. Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 59–66. IEEE.

Mysore, S.; Mabsout, B.; Mancuso, R.; and Saenko, K. 2021. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 1810–1816. IEEE.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shang, W.; Wang, X.; Srinivas, A.; Rajeswaran, A.; Gao, Y.; Abbeel, P.; and Laskin, M. 2021. Reinforcement learning with latent flow. *Advances in Neural Information Processing Systems*, 34: 22171–22183.

Shen, Q.; Li, Y.; Jiang, H.; Wang, Z.; and Zhao, T. 2020. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, 8707–8718. PMLR.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Stooke, A.; Lee, K.; Abbeel, P.; and Laskin, M. 2021. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, 9870–9879. PMLR.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

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

Yarats, D.; Kostrikov, I.; and Fergus, R. 2020. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International conference on learning representations*.

Yu, T.; Lan, C.; Zeng, W.; Feng, M.; Zhang, Z.; and Chen, Z. 2021. Playvirtual: Augmenting cycle-consistent virtual trajectories for reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 5276–5289.

Yu, T.; Zhang, Z.; Lan, C.; Lu, Y.; and Chen, Z. 2022. Mask-based latent reconstruction for reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 25117–25131.

Zagoruyko, S.; and Komodakis, N. 2016. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*.

Zeng, A.; Song, S.; Lee, J.; Rodriguez, A.; and Funkhouser, T. 2020. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4): 1307–1319.

Zhai, Y.; Peng, P.; Zhao, Y.; Huang, Y.; and Tian, Y. 2023. Stabilizing Visual Reinforcement Learning via Asymmetric Interactive Cooperation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 207–216.

Zhang, A.; McAllister, R.; Calandra, R.; Gal, Y.; and Levine, S. 2020. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*.

Zheng, R.; Wang, X.; Sun, Y.; Ma, S.; Zhao, J.; Xu, H.; Daumé III, H.; and Huang, F. 2024. TACO: Temporal Latent Action-Driven Contrastive Loss for Visual Reinforcement Learning. *Advances in Neural Information Processing Systems*, 36.

Zhu, J.; Xia, Y.; Wu, L.; Deng, J.; Zhou, W.; Qin, T.; Liu, T.-Y.; and Li, H. 2022. Masked contrastive representation learning for reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3): 3421–3433.

Zhu, Y.; Wong, J.; Mandlekar, A.; Martín-Martín, R.; Joshi, A.; Nasiriany, S.; and Zhu, Y. 2020. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*.