Self-Predictive Dynamics for Generalization of Vision-based Reinforcement Learning

Kyungsoo Kim, **Jeongsoo Ha** and **Yusung Kim***
Sungkyunkwan University
{unigary, hjg1210}@g.skku.edu, yskim525@skku.edu

Abstract

Vision-based reinforcement learning requires efficient and robust representations of image-based observations, especially when the images contain distracting (task-irrelevant) elements such as shadows, clouds, and light. It becomes more important if those distractions are not exposed during training. We design a Self-Predictive Dynamics (SPD) method to extract task-relevant features efficiently, even in unseen observations after training. SPD uses weak and strong augmentations in parallel, and learns representations by predicting inverse and forward transitions across the two-way augmented versions. In a set of MuJoCo visual control tasks and an autonomous driving task (CARLA), SPD outperforms previous studies in complex observations, and significantly improves the generalization performance for unseen observations. Our code is available at https://github.com/unigary/SPD.

1 Introduction

Vision-based reinforcement learning (RL) [Hafner *et al.*, 2019; Srinivas *et al.*, 2020; Zhang *et al.*, 2020] has been studied to learn optimal control using high dimensional image inputs. The demand for vision-based RL has continued to grow as more attempts are made to apply RL to real-world applications such as robotics and autonomous driving, which primarily use image data. However, to achieve this, vision-based RL must address two fundamental problems; data efficiency and generalization. Data efficiency refers to how quickly optimal control of a task can be learned using fewer experience samples. Learning control from high dimensional images such as raw pixels inevitably increases the learning difficulty. In particular, if the images contain task-irrelevant information (clouds, shadows, and light etc.), this unnecessary information interferes with learning optimal control. The more complex the observation, the worse this problem is. In terms of generalization, task-irrelevant information may vary depending on the time and location of the actual tests. If those distracting elements are not exposed during training, control performance could be severely degraded. Some prior works present that using relatively weak data augmentations can improve data efficiency rather than using strong augmentations [Laskin *et al.*, 2020]. However, we found that it is not sufficient if the observed characteristics at the time of testing differ from those at the time of training as shown in Table 1.

In this work, we design Self-Predictive Dynamics (SPD) as a method of self-supervised learning suitable for vision-based RL. Our method introduce two-way data augmentations which apply both weak and strong augmentation techniques for the same observation. First, we use a discriminator to distinguish between two-way augmented observations, while our encoder learns to fool the discriminator. It helps that our encoder to capture invariant features from the different-level augmented versions. Second, SPD infers actually conducted actions between successive (latent) states across two-way augmentations. The inferred actions are used to predict the identical future states from (two-way augmented) current states. By accurately understanding dynamics chaining (from inverse to forward dynamics), SPD can learn optimal control policies more efficiently in complex visual environments, and shows excellent generalization performance especially for unseen observations. For evaluation, we used a set of continuous control tasks (the DeepMind Control suite [Tassa *et al.*, 2018]) with distracting elements backgrounds as proposed in [Zhang *et al.*, 2020]. Compared to prior studies, SPD efficiently learns a control policy in both simple and complex observations. We also show that SPD significantly outperforms existing studies when the testing observations differ from the training observations, which means higher generalization ability. In an autonomous driving task, CARLA [Dosovitskiy *et al.*, 2017], our method achieves the best performance on complex observations containing a lot of task-irrelevant information in realistic driving scenes.

The key contributions of this paper are as follows:

- We introduce a Self-Predictive Dynamics (SPD) method using both weak and strong augmentations in parallel. SPD enables one-stage learning generalization without additional pre-training or fine-tuning processes.
- SPD outperforms previous studies in complex backgrounds, and shows the best generalization performance when observed characteristics change in real-world scenarios after training.

*Corresponding author

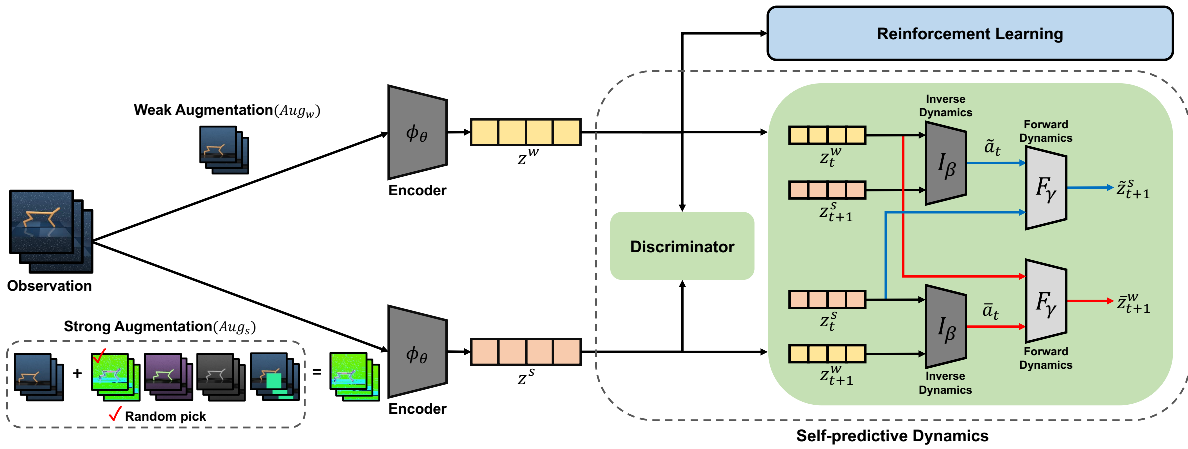

Figure 1: Our Framework Overview: we use a shared encoder for RL and Self-Predictive Dynamics (SPD). An observation is augmented in two ways; Aug_w uses Random-Shift only, and Aug_s uses Random-Shift and other randomly chosen augmentation method. The encoded latent state z^w is used to train an RL algorithm, and both z^w and z^s are passed to SPD.

2 Related Works

In vision-based RL studies, representation learning is a fundamental component for achieving an optimal control policy. Many studies have been conducted on improving data efficiency and generalization, mainly using data augmentation techniques and self-supervised learning methods.

2.1 Data Efficiency in Vision-based RL

Some studies have introduced pixel-level reconstructions for representation learning using variational inference [Yarats *et al.*, 2019; Lee *et al.*, 2019a]. By reconstructing the current observation accurately, it helps to extract compact representations of image observations. It has been shown that learning forward dynamics to predict the future state [Schwarzer *et al.*, 2020; Oord *et al.*, 2018] can be effective for making better representations. Several RL studies proposed to use data augmentations which provide different views of the image data. RAD [Laskin *et al.*, 2020] has shown that using data augmentations improves data efficiency without modifying RL algorithms. DrQ [Kostrikov *et al.*, 2020] has improved data efficiency using both data augmentation methods and modified Q-functions. CURL [Srinivas *et al.*, 2020] has combined data augmentations and contrastive learning [Chen *et al.*, 2020] to learn representation more efficiently. These studies have used relatively simple and weak such as random-crop or random-shift. Although those weak augmentations are useful to improve data efficiency in simple backgrounds, they are NOT working well for complex or unseen observations.

2.2 Generalization in Vision-based RL

In vision-based control tasks, not only image observations include information not relevant to the task such as clouds, shadows, and light, but these distracting factors can change continuously over the duration of the test. Therefore, extracting invariant features relevant to the task control is a key challenge for improving generalization. DBC [Zhang *et al.*, 2020] used bisimulation metrics to provide effective downstream control by learning invariant features from the images including task-irrelevant details. DBC shows the potential for generalization, but the performance achieved is still low. Inverse dynamics has been used as one of self-supervised auxiliary tasks in RL [Pathak *et al.*, 2017]. PAD [Hansen *et al.*, 2020] has used inverse dynamics with weak data augmentations not only training a policy but fine-tuning to adapt the policy to new environments. Some recent studies have suggested the use of strong data augmentation techniques that heavily distort the image such as Color-jitter or Random-convolution [Lee *et al.*, 2019b]. Strong augmentations are known to lead to robust and generalizable representations for vision research areas, but naively applying them into RL results in sub-optimal performance [Laskin *et al.*, 2020]. SODA [Hansen and Wang, 2021] learns representation by maximizing the mutual information between strong augmented data and non-augmented data. SECANT [Fan *et al.*, 2021] first learns an expert policy with weak augmentations, and imitates the expert policy with strong augmentations.

Our work suggests Self-Predictive Dynamics (SPD) across two-way (weak and strong) data augmentations in parallel. The learning process of SPD is simple and does NOT require any pre-training or fine-tuning after deployments.

3 Self-Predictive Dynamics

In this section, we introduce Self-Predictive Dynamics (SPD) which consists of the two-way data augmentations, discriminator and dynamics chaining. Our method does not require any changes to the underlying RL algorithm, and any RL algorithm can be used.

3.1 Model Overview

We design the model architecture to share represented features that feed into SPD and RL. We define encoder ϕ, discriminator D, and dynamics chaining ψ. Our goal is to train the encoder ϕ to extract task-control relevant information efficiently so that the RL agent can learn the generalized optimal policy. The encoder ϕ is updated with the gradients of SPD and RL. The model overview is illustrated in Figure 1 and Algorithm 1.

3.2 Two-way Data Augmentations

We introduce a two-way data augmentation method. The weak and strong augmented versions are used in parallel during training. **Random-shift** [Kostrikov *et al.*, 2020] is used for a weak augmentation technique. It pads each side and then selects a random crop back to the original image size. For strong augmentation techniques, we use a combination of Random-shift and a randomly chosen one among the following four techniques. **Grayscale** converts RGB images to grayscale images based on certain probabilities. **Random convolution** [Lee *et al.*, 2019b] transforms an image through a randomly initialized convolutional layer. **Color-jitter** converts RGB image to HSV image which adds noise to each channel of HSV. **Cutout-color** [Cobbe *et al.*, 2019] randomly inserts a small random color occlusion into the input image.

In Figure 1, two-way augmentations are shown for a given observation. Aug_w stands for a weak augmented version and Aug_s represents a strong augmented version. In our ablation test, using multiple strong augmentation techniques together shows better performance than using a single strong augmentation in the supplementary material.

3.3 Discriminator

The goal of the discriminator is for the encoder to reduce the difference between the representations for the weak and strong augmented versions. When two-way data augmentations Aug_w and Aug_s pass through the encoder ϕ, it produces latent states $z^w = \phi(Aug_w(obs))$ and $z^s = \phi(Aug_s(obs))$ where obs is an image observation. For the discriminator, we use the concept of a relativistic GAN [Jolicoeur-Martineau, 2018], which is known to be more stable and faster than a standard GAN. For z^w and z^s, we define encoder (as a generator) and discriminator objective functions as follows, where σ represents a sigmoid function.

$$J(\phi) = -\log(\sigma(D(z^s) - D(z^w))), \tag{1}$$

$$J(D) = -\log(\sigma(D(z^w) - D(z^s))). \tag{2}$$

$J(\phi)$ optimizes z^s to have a higher value than z^w in Equation 1. Conversely, $J(D)$ optimizes z^w to have a higher value than z^s in Equation 2. By alternately optimizing Equation 1 and Equation 2, the encoder ϕ is updated so that the representations of z^w and z^s become similar. Eventually, our discriminator helps to learn invariant features regardless of the position shifts and the changes in color and texture of the observations.

3.4 Dynamics Chaining

We introduce a dynamics chaining which consists of inverse dynamics and forward dynamics based on two-way data augmentations. For given sequential latent states z_t^w and z_{t+1}^s, and another pair of z_t^s and z_{t+1}^w, inverse dynamics I infers the actions $\tilde{a}_t = I(z_t^w, z_{t+1}^s)$ and $\bar{a}_t = I(z_t^s, z_{t+1}^w)$. Even if the input images are augmented with different levels, the two inferred actions should be similar to each other, and should be nearly identical to the action a_t actually performed.

Algorithm 1 Self-Predictive Dynamics

Initialize: Encoder ϕ, Policy π, Critic Q, Discriminator D, Dynamics chaining ψ, Buffer B.

```
for each iteration do
    for each environment step do
        Encode state z_t = φ(s_t)
        Execute action a_t = π(z_t)
        Store transition: B ← B ∪ {s_t, a_t, s_{t+1}, r_t}
    end for
    for each update step do
        Sample mini-batch: (S, A, S', R) ~ B
        // Apply weak augmentation
        Z_w, Z'_w = Aug_w(S), Aug_w(S')
        // Apply strong augmentation
        Z_s, Z'_s = Aug_s(S), Aug_s(S')
        // Train self-supervisions
        E_{Z_w, Z'_w, Z_s, Z'_s, A}[J(ψ, φ, D)]
        // Train RL Policy
        E_{Z_w, Z'_w}[J(π)]
    end for
end for
return Optimal Policy π
```

The inferred actions $\tilde{a}_t$ and $\bar{a}_t$ are fed into forward dynamics F along with the current latent states z_t^s and z_t^w. F predicts the next latent states as following; $\tilde{z}_{t+1}^s = F(z_t^s, \tilde{a}_t)$ and $\bar{z}_{t+1}^w = F(z_t^w, \bar{a}_t)$. $\tilde{z}_{t+1}^s$ and $\bar{z}_{t+1}^w$ are predicted across two-way augmented versions in parallel, they should be identical to z_{t+1}^s and z_{t+1}^w. This dynamics chaining allows our encoder to learn more powerful representations by using both dynamics knowledge inferred across two-way augmented observations.

The inverse dynamics objective function Equation 3 is defined as the mean squared error between actual action and inferred action.

$$J(I) = \frac{(I(z_t^w, z_{t+1}^s) - a_t)^2 + (I(z_t^s, z_{t+1}^w) - a_t)^2}{2} \tag{3}$$

The forward dynamics objective function Equation 4 is defined as negative cosine similarity Δ between the predicted next latent state and the actual next latent state that encodes the next observation.

$$J(F) = \frac{\Delta(\tilde{z}_{t+1}^s, z_{t+1}^s) + \Delta(\bar{z}_{t+1}^w, z_{t+1}^w)}{2} \tag{4}$$

The dynamics chaining objective function Equation 5 is defined as a combination of inverse dynamics and forward dynamics.

$$J(\psi) = J(I) + J(F) \tag{5}$$

The Self-Predictive Dynamics (SPD) objective function is defined as a combination of dynamics chaining and discriminator as shown in Equation 6, and it can send a training signal to the encoder ϕ to efficiently represent task-relevant features.

$$J(\psi, \phi, D) = \lambda_\psi J(\psi) + \lambda_A J(\phi, D) \tag{6}$$

where λ_ψ and λ_A are hyper parameters.[1]

[1] We have shown that SPD has good performance in a wide range of hyper parameter choices in the supplementary material.

	SAC	DrQ	CURL	SODA	PAD	SPD (Ours)
Data Efficiency (training and testing on Simple Distractor)						
Cheetah Run	230.2±20.4	272.8±31.4	**335.5±0.3**	304.2±23.7	301.0±32.9	333.8±2.5
Finger Spin	399.7±25.3	665.1±27.4	656.2±47.8	735.9±33.7	689.7±27.7	**983.9±0.7**
Hopper Hop	92.4±5.6	91.5±31.8	73.6±27.5	86.4±43.5	125.3±86.5	**152.5±6.0**
Reacher Easy	107.3±0.4	230.2±47.4	409.2±45.0	286.4±50.0	286.7±160.4	**645.5±107.1**
Walker Walk	37.1±4.5	493.5±105.2	**917.4±12.0**	869.1±12.0	861.7±1.8	895.0±7.3
Data Efficiency (training and testing on Natural Video)						
Cheetah Run	136.4±22.4	63.8±19.7	118.2±38.2	74.0±31.0	171.0±113.8	**330.2±25.5**
Finger Spin	288.8±11.9	205.0±144.5	227.4±146.9	58.7±40.0	3.3±1.7	**983.2±1.2**
Hopper Hop	33.1±7.1	0.0±0.0	9.7±5.4	0.3±0.2	0.7±0.6	**164.3±14.1**
Reacher Easy	100.1±1.9	89.7±14.6	413.9±106.7	80.4±4.8	104.0±20.5	**574.4±61.9**
Walker Walk	32.7±2.4	104.4±43.3	811.9±52.1	404.1±47.3	72.5±7.7	**895.8±17.9**
Generalization (training on Simple Distractor but testing on Natural Video)						
Cheetah Run	51.0±18.4	218.6±25.2	189.5±31.1	228.7±17.8	298.3±28.7	**328.7±6.2**
Finger Spin	125.2±27.0	661.3±26.8	647.2±44.4	652.5±38.5	690.0±27.7	**893.2±29.5**
Hopper Hop	14.8±5.2	81.4±30.0	42.7±23.6	59.2±32.0	112.7±67.7	**134.6±6.2**
Reacher Easy	109.4±3.6	158.3±20.2	286.9±46.7	160.4±28.0	273.7±158.0	**431.5±118.8**
Walker Walk	57.8±16.9	270.5±81.6	407.6±35.0	754.2±25.4	835.3±1.4	**854.6±16.3**

Table 1: Performance of SPD and baselines on five tasks in the DeepMind Control suite. We train for 500K environment steps on *Simple Distractor* and *Natural Video*. We evaluate the trained model on the same *Simple Distractor* and *Natural Video* for data efficiency experiments, and evaluate the model which is trained on *Simple Distractor* on unseen *Natural Video* for generalization experiments. The results show the mean and standard deviation over three different seeds.

Algorithm 1 describes how SPD works. In the algorithm, s_t, s_{t+1} are the image observations obtained by interacting with the environment. We divide the training phase of SPD into two steps. First, train an encoder by optimizing SPD objective, and then train the RL policy. We repeat this learning process and SPD objective functions refer to Equation 6. This algorithm version is based on an off-policy RL algorithm, such as Soft Actor-Critic (SAC) [Haarnoja *et al.*, 2018], but our method (SPD) can work with any RL algorithms, as shown in the supplementary material. (such as on-policy algorithms like PPO [Schulman *et al.*, 2017] and other off-policy algorithms like TD3 [Fujimoto *et al.*, 2018]).

4 Experiments

This section demonstrates how efficiently SPD can learn vision-based control tasks with distracting elements (task-irrelevant information) and can generalize well against unseen test environments. On a set of continuous control tasks in the DeepMind Control suite, SPD shows excellent performance in most settings. For CARLA [Dosovitskiy *et al.*, 2017], a more realistic and autonomous driving environment with various distractors (e.g., shadows, changing weather, and light), we also show better performance than prior studies. We benchmark SPD against the following algorithms; SAC is plain Soft Actor-Critic with no augmentation. DrQ [Kostrikov *et al.*, 2020] applies data augmentations and regularized Q-function in SAC. CURL [Srinivas *et al.*, 2020] introduces a method of combining contrastive representation learning and RL. SODA [Hansen and Wang, 2021] learns representation by maximizing the mutual information between augmented and non-augmented data. PAD [Hansen *et al.*, 2020] fine-tunes representations at testing environments through self-supervision.

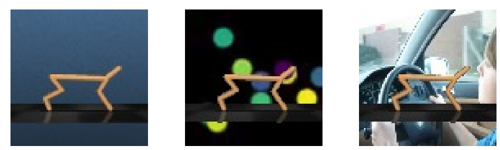

Figure 2: We use three different background types. There are examples on a Cheetah task in the Deepmind Control suite; *Default* (left), *Simple Distractor* (center), and *Natural Video* (right)

4.1 Network Architecture

We implement our SPD on top of Soft Actor Critic (SAC) for the visual input version [Yarats *et al.*, 2019] architecture, which updates the encoder only with Q-function back-propagation. The RL parts Actor, Critic and the self-supervised part SPD share the Encoder ϕ which consists of 4 convolutional layers and 1 fully connected layer. Both Actor and Critic consists of 3 fully connected layers. Dynamics chaining ψ and Discriminator D consists of 4 fully connected layers and 2 fully connected layers, respectively. For CARLA, we modify the Encoder ϕ slightly. Implementation details and hyper parameters are in the supplementary material.

4.2 DeepMind Control Suite

The DeepMind Control suite is a vision-based simulator that provides a set of continuous control tasks. We experiment with nine tasks; Cheetah Run, Finger Spin, Hopper Hop, Reacher Easy, Walker Walk and additional tasks in the supplementary material. And we evaluate the performances on

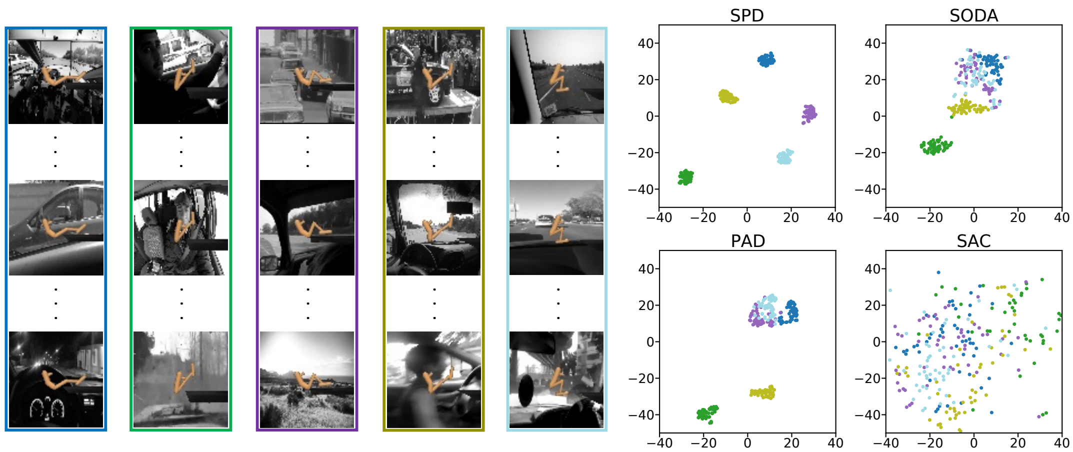

Figure 3: t-SNE of representations learned by SPD, SODA, PAD and SAC. Even if the background is dramatically different, SPD can encode behaviorally-equivalent observations (blue, green, violet, olive, sky blue) to be most closely located.

two metrics; one is Data efficiency and the other is Generalization. Each RL method is trained for 500K environment steps, and every 5,000 steps, we evaluated the currently trained model by calculating the average return for 10 episodes. We trained each RL method over three different seeds. As shown in Table 1, SPD shows performance similar to the best performance of the prior works in data efficiency experiments with lower distractions, but significantly outperforms the prior works in data efficiency experiments with higher distractions and generalization experiments. More experiment details are in the supplementary material.

Data Efficiency

For the data efficiency evaluation, we used two background configurations; *Simple Distractor* and *Natural Video*, as shown in Figure 2. *Simple Distractor* is a non-stationary background with randomly plotted circles with different colors. *Natural Video* is also a non-stationary background which consists of real car-driving scenes in Kinetics dataset [Kay *et al.*, 2017]. In this evaluation, the test is carried out in the same environment (the same background setup) used for training. Basically, the higher the level of distraction, the lower the task performance. As shown in Table 1, SPD outperforms other baselines on **3** out of **5** tasks in the *Simple Distractor* background, but **5** out of **5** tasks in the *Natural Video* background. For example, SPD achieves performance gains of **22%** and **396%** on the Hopper Hop, compared to the best performance among the other RL methods for each background setup. The learning curves for task environments and the additional backgrounds are in the supplementary material.

Generalization

In this experiment, we first trained each RL method in the *Simple Distractor* background and then evaluated it in the *Natural Video* background, which was not seen during the training phase. The bottom row in Table 1 presents that the generalization performance for unseen observations. SPD significantly outperforms other baselines for all environments. On Finger Spin, SPD achieves **29%** higher performance than PAD which is fine-tuned for testing observations. All nine environments results and their learning curves are provided in the supplementary material. In Figure 3, we also visualize the state embedding of Hopper Hop using t-SNE. Even if unseen backgrounds are dramatically different, a well-generalized encoder should capture invariant features when observations are behaviorally equivalent. It has been shown that SPD can encode semantically similar observations to be most closely located.

Ablation Studies

We present the ablation studies to examine the synergy of our two-way data augmentations, discriminator, and dynamics chaining. Our ablation experiment is conducted in the same environment setup as the Generalization experiment. In Figure 4 (left), Discriminator Only stands for SAC with two-way data augmentations and the discriminator but no dynamics chaining. Discriminator + Inverse consists of two-way data augmentations, the discriminator, and the inverse dynamics (without the forward dynamics). The performance of Discriminator Only shows the lowest generalization performance. Discriminator + Inverse shows the performance can be highly improved because of the inverse dynamics. Although the role of the inverse dynamics greatly affects the performance, there is no doubt that our full integration (SPD) achieves the best performance.

In another ablation test, we try to analyze the role of the discriminator. We compare SPD to a version without the discriminator and a version with the contrastive learning method [Srinivas *et al.*, 2020]. Figure 4 (right) shows clear differences in achieved task performance according to the different discriminator settings. Using the contrastive learning method improves performance compared to no-discriminator version. However, SPD using a relativistic GAN as the discriminator outperforms the version using the contrastive learning method much better.

4.3 CARLA Environment

CARLA is a first-person view simulator for studying autonomous driving systems. In the CARLA simulations, we

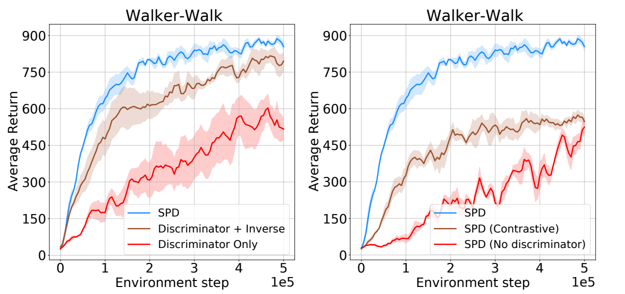

Figure 4: (left) Ablation studies for SPD. We test the effects of discriminator, inverse dynamics and dynamics chaining. (right) Ablation studies for the different discriminator types. We show each ablation studies on three different seeds with 1.0 standard error shaded.

Figure 5: Scenes in CARLA simulations classified as Highway (left column), Town (center column) and Bridge (right column). Each column is captured in the same spot but contains different task-irrelevant information such as the Sun, rain, shadows, clouds, etc.

can evaluate the performance of RL methods on more realistic visual observations. As shown in Figure 5, there are diverse types of distractors (e.g., the Sun, rain, shadows, clouds, etc.) around the agent, and it changes dynamically with every episode, and even within the same episode. Therefore, it becomes more important to extract control-related features (e.g., road, collision, speed, brake, steer, etc.). The basic experimental setup is configured the same as DBC [Zhang *et al.*, 2020]. Visual observation is a 300 degree view from the vehicle roof and the image size is $3{\times}84{\times}420$. The reward is defined by the function of driving distance, speed, and the penalty of collision, steering and breaking. Each method is trained for 100K environment steps, and the average return for 20 test episodes is calculated. We run each RL method across three seeds. Figure 6 shows the performance comparison with three seeds in CARLA. SODA performs better than other baselines and is comparable with the performance of SPD, but SPD learns much faster and achieves the highest performance. For another comparison of representation quality, we suggest the representation distance in latent space between two observations. We can intuitively assume that the representation distance should be close if their task-relevant context is similar regardless of other distracting elements. We first took 50 random observations at three locations; Highway, Town, and Bridge in CARLA. We repeatedly collected observations from almost the same spots, but these observation characteristics change because of varying task-irrelevant

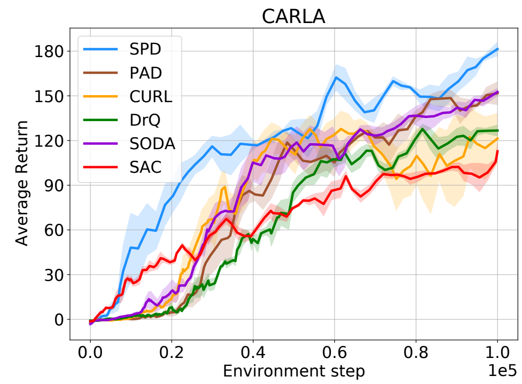

Figure 6: Performance comparison in the autonomous driving environment CARLA. SPD outperforms all other baselines.

	SAC	DrQ	CURL	SODA	PAD	SPD
Highway	3.86	2.20	2.38	1.74	1.83	**1.00**
Town	6.23	3.57	3.86	2.43	2.47	**1.00**
Bridge	3.82	1.61	1.57	1.25	1.41	**1.00**

Table 2: Average representation distance of latent space according to task-irrelevant information changes in CARLA simulations. (The numbers are normalized to SPD)

information (e.g., the Sun, shadows, clouds, rain, car types & colors, etc.), as shown in Figure 5. We measured the L2 distance in the latent space between various observations obtained under behaviorally identical circumstances. Table 2 presents the average representation distance normalized to the SPD result. It shows that SPD has minimal average distance compared to other studies, and we believe this is why our method performs best.

5 Conclusion

In this work, we propose a novel representation learning method for vision-based RL. Our proposed Self-Predictive Dynamics based on two-way (weak and strong) data augmentations can significantly improve the data efficiency and generalization performance when operating on highly complex or unseen observations. In the future, we plan to design a sequence-based generalization approach such as representing a series of image inputs and predicting multi-step dynamics chaining in latent space.

Acknowledgements

This work was supported partly by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government (MSIT) (No. 2022-0-01045, Self-directed Multi-Modal Intelligence for solving unknown, open domain problems), (No. 2022-0-00688, AI Platform to Fully Adapt and Reflect Privacy-Policy Changes), (No. 2020-0-00973, Reconstruction of Non-Line-of-Sight Scene for VR/AR Contents) and (No. 2019-0-00421, Artificial Intelligence Graduate School Program(Sungkyunkwan University)).

References

[Chen *et al.*, 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[Cobbe *et al.*, 2019] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

[Dosovitskiy *et al.*, 2017] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[Fan *et al.*, 2021] Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Anima Anandkumar. Secant: Self-expert cloning for zero-shot generalization of visual policies. *arXiv preprint arXiv:2106.09678*, 2021.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[Haarnoja *et al.*, 2018] 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.

[Hafner *et al.*, 2019] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[Hansen and Wang, 2021] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. IEEE, 2021.

[Hansen *et al.*, 2020] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

[Jolicoeur-Martineau, 2018] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

[Kay *et al.*, 2017] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[Kostrikov *et al.*, 2020] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[Laskin *et al.*, 2020] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[Lee *et al.*, 2019a] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

[Lee *et al.*, 2019b] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *arXiv preprint arXiv:1910.05396*, 2019.

[Oord *et al.*, 2018] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[Pathak *et al.*, 2017] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Schwarzer *et al.*, 2020] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

[Srinivas *et al.*, 2020] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[Tassa *et al.*, 2018] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[Yarats *et al.*, 2019] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

[Zhang *et al.*, 2020] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.