

# APPENDIX

## A Supplemented Details of BCV-LR Methodology

### A.1 Pseudo Code

---

**Algorithm 1** The pseudo code of the proposed BCV-LR.

---

**Require:** The action-free video dataset  $B^v$ , the reward-free environment  $E$ , the latent feature pre-training update times  $U_{lf}$ , latent action pre-training steps  $U_{la}$ , the online environmental interaction steps  $I_{ft}$ , the online finetuning frequency  $F_{ft}$ , the finetuning update times  $U_{ft}$ , the behavior cloning update times  $U_{bc}$ .

**Initialize:** The latent feature encoder  $f$ , the latent action predictor  $p$ , the world model  $w$ , the latent action decoder  $d$ , the latent policy  $\pi$ , and the environmental replay buffer  $B^e$ .

```
1: # Offline stage #
2: ## Pre-training latent features ##
3: for  $index = 1, \dots, U_{lf}$  do
4:   Sample a batch of video observations from video dataset  $B^v$ .
5:   Calculate the self-supervised objective  $\mathcal{L}_{lf}$  via Equation (1).
6:   Update the latent feature encoder  $f$  by backpropagation.
7: ## Pre-training latent actions ##
8: for  $index = 1, \dots, U_{la}$  do
9:   Sample a batch of video observations from video dataset  $B^v$ .
10:  Calculate the dynamics-based objective  $\mathcal{L}_{la}$  via Equation (2).
11:  Update the latent action predictor  $p$  and the world model  $w$  by backpropagation.
12: # Online stage #
13: for  $index = 1, \dots, I_{ft}/F_{ft}$  do
14:   ## Optional preparations ##
15:   (Optional, letting latent policy  $\pi$  fully imitate the pre-trained latent actions before interactions.)
16:   if  $index == 1$  :
17:     Conduct latent behavior cloning via Equation (4) until convergence.
18:   ## Collect environmental interactions ##
19:   for  $index' = 1, \dots, F_{ft}$  do
20:     if  $index == 1$  :
21:       Interact with environment  $E$  to collect data with random policy.
22:       Save the data into the buffer  $B^e$ 
23:     else:
24:       Interact with environment  $E$  to collect data with the learned policy (consisting of  $f, \pi$ , and  $d$ ).
25:       Save the data into the buffer  $B^e$ 
26:   ## Finetuning and decoding the latent actions ##
27:   for  $index' = 1, \dots, U_{ft}$  do
28:     Sample a batch of environmental transitions from buffer  $B^e$ .
29:     Calculate the latent action finetuning objective  $\mathcal{L}_{ft}$  via Equation (3).
30:     Update the latent action predictor  $p$  and the latent action decoder  $d$  by backpropagation.
31:   ## Behavior cloning the latent policy ##
32:   for  $index' = 1, \dots, U_{bc}$  do
33:     Sample a batch of action-free observations from videos  $B^v$ .
34:     Obtain the predicted latent actions through  $f$  and  $p$ .
35:     Calculate the latent behavior cloning objective  $\mathcal{L}_{bc}$  via Equation (4).
36:     Update the latent policy  $\pi$ .
```

**Output:** The well-trained visual control policy ( $f, \pi$ , and  $d$ ).

---

## A.2 Prototype-based Temporal Association Task

As shown in Section 3.1.1, for partially observable domains with complex dynamics, such as DMControl benchmark [31] in this paper, we employ a recent advanced prototype-based temporal association task [57, 43, 34], which aligns two temporally neighboring observations together in the latent space. For the sake of fluency, we put the details of this self-supervised task here instead of in the main text.

Concretely,  $M$  observation pairs  $\{(o_i^v, o_{i+1}^v)\}$  are sampled from videos, augmented by the random shift, and encoded by  $f$  and  $f'$  to obtain latent features  $\{s_i^v\}$  and  $\{s_{i+1}^v\}$ . Each current latent feature  $s_i^v$  is further processed by an MLP  $u$ , which is set to introduce asymmetry for avoiding collapse to trivial solutions. Then, we take a softmax over the dot product between  $u(s_i^v)$  and  $M$  trainable prototypes  $\{c_j\}_{j=1}^M$ :

$$x_i^v = \text{softmax} \left( \frac{u(s_i^v) \cdot c_1}{\tau}, \dots, \frac{u(s_i^v) \cdot c_M}{\tau} \right), \quad (5)$$

where  $\tau$  denotes a temperature hyper-parameter. To calculate the association target, the Sinkhorn-Knopp [61] algorithm is employed on the whole batch  $\{s_{i+1}^v\}$  and prototypes  $\{c_j\}$  to obtain batch-clustering target labels  $\{y_{i+1}^v\}$  for all training feature pairs. Concretely, the Sinkhorn-Knopp algorithm begins with the square matrix  $C$ , whose elements are computed by the dot product over each  $s_{i+1}^v$  and prototype  $c_j$ :

$$C_{ij} = s_{i+1}^v \cdot c_j. \quad (6)$$

Then it employs several times of doubly-normalization on the matrix  $C$  to obtain the clustering target matrix  $T$ , constraining every column and row to have the same sum with as little change of original  $C$  as possible. One doubly-normalization consists of a row normalization and a column normalization. The row normalization and column normalization are formulated as the following:

$$\text{NormRow}(C) = \frac{1}{M} \text{diag}(\text{SumRow}(C)^{-1}) \cdot C, \quad (7)$$

$$\text{NormColumn}(C) = \frac{1}{M} C \cdot \text{diag}(\text{SumColumn}(C)^{-1}), \quad (8)$$

where  $\text{SumRow}(\cdot)$  denotes the row addition,  $\text{SumColumn}(\cdot)$  denotes the column addition, and  $\text{diag}(\cdot)$  denotes the diagonalization of a matrix. The doubly-normalization,  $\text{NormDouble}(\cdot)$ , is correspondingly defined as:

$$\text{NormDouble}(C) = \text{NormColumn}(\text{NormRow}(C)). \quad (9)$$

Several times of doubly-normalization are applied to  $C$  to obtain the clustering target matrix  $T$ . The  $i$ -th row of  $T$  is the clustering-based target  $y_{i+1}^v$  of the  $x_i^v$ . The self-supervised objective is formulated as:

$$L_{lf} = -y_{i+1}^v{}^\top \log x_i^v. \quad (10)$$

The clustering-based temporal association task actually makes each observation access its own temporally neighboring observation in the self-supervised latent space, thus enabling the representation module  $f$  to obtain the temporal information.

## B Additional Results

### B.1 Numerical Results at Fewer Steps in DMControl

In Section 4.3, we provide the results at 100k environmental steps, where the proposed BCV-LR leads across 6/8 tasks. Here we demonstrate the advantages of BCV-LR when interactions are further constrained (50k and 20k steps). As shown in Table 8 and Table 9, BCV-LR performs much better than both ILV and RL baselines, enabling effective policy learning given only 20k steps.

Table 4: Numerical results at 50k environmental steps. *Italics* indicate using expert videos, and underlines denote using environmental rewards. **Bold text** indicates the highest score excluding video experts.

Task (50k steps)	<i>BCV-LR (ours)</i>	<i>LAIFO</i> [50]	<i>BCO</i> [26]	<i>UPESV</i> [28]	<u>TACO</u> [60]	<u>DrQv2</u> [3]	Expert Videos
point_mass_easy	<b>743 ± 76</b>	1 ± 0	3 ± 4	3 ± 4	529 ± 85	107 ± 143	885
reacher_hard	<b>860 ± 64</b>	22 ± 33	261 ± 142	14 ± 6	41 ± 44	4 ± 3	967
jaco_reach_top_right	<b>73 ± 13</b>	14 ± 14	0 ± 0	3 ± 3	10 ± 8	22 ± 16	198
finger_spin	<b>912 ± 38</b>	166 ± 122	855 ± 28	0 ± 0	416 ± 41	165 ± 121	981
ball_in_up_catch	<b>683 ± 106</b>	25 ± 43	208 ± 227	33 ± 47	126 ± 115	176 ± 120	986
jaco_reach_bottom_left	<b>101 ± 32</b>	20 ± 22	0 ± 0	8 ± 4	8 ± 6	12 ± 10	203
cheetah_run_backward	268 ± 8	1 ± 0	207 ± 27	3 ± 1	<b>272 ± 157</b>	220 ± 158	389
reacher_easy	<b>747 ± 101</b>	80 ± 17	110 ± 123	3 ± 4	110 ± 15	202 ± 152	975

Table 5: Numerical results at 20k environmental steps. *Italics* indicate using expert videos, and underlines denote using environmental rewards. **Bold text** indicates the highest score excluding video experts.

Task (20k steps)	<i>BCV-LR (ours)</i>	<i>LAIFO</i> [50]	<i>BCO</i> [26]	<i>UPESV</i> [28]	<u>TACO</u> [60]	<u>DrQv2</u> [3]	Expert Videos
point_mass_easy	<b>318 ± 219</b>	1 ± 1	0 ± 0	0 ± 0	1 ± 0	1 ± 1	885
reacher_hard	<b>714 ± 46</b>	22 ± 36	69 ± 85	20 ± 14	26 ± 35	8 ± 8	967
jaco_reach_top_right	<b>70 ± 22</b>	5 ± 5	6 ± 8	4 ± 3	2 ± 1	3 ± 2	198
finger_spin	<b>944 ± 25</b>	84 ± 84	799 ± 82	0 ± 0	51 ± 68	43 ± 59	981
ball_in_up_catch	<b>459 ± 139</b>	96 ± 121	196 ± 215	99 ± 81	99 ± 81	33 ± 47	986
jaco_reach_bottom_left	<b>45 ± 28</b>	1 ± 1	0 ± 1	8 ± 6	3 ± 3	2 ± 2	203
cheetah_run_backward	<b>226 ± 11</b>	1 ± 0	188 ± 14	5 ± 1	150 ± 94	67 ± 48	389
reacher_easy	<b>569 ± 206</b>	77 ± 32	131 ± 92	0 ± 0	136 ± 59	38 ± 31	975

## B.2 Experiments in Metaworld Manipulation

In this section, we further conduct some extra experiments in the Metaworld manipulation benchmark [32], which may demonstrate a wider application of BCV-LR. For each Metaworld task, only 50k environmental steps are allowed. The remaining settings are similar to that of DMControl experiments, except for the self-supervised latent feature objective  $\mathcal{L}_{lf}$  employed in BCV-LR. Concretely, we employ only contrastive learning loss (described in Section 3.1.1) in this benchmark because it seems to perform better than other objectives used in this paper. Results (success rate) are shown in Table 6. In this interaction-limited situation, BCV-LR can still derive effective manipulation skills from expert videos without accessing expert actions and rewards, which demonstrates its wider range of applications and potential for generalizing to real-world manipulation tasks.

Table 6: The results (success rate) on 4 manipulation tasks from Metaworld. *Italics* indicate using expert videos, and underlines denote using environmental rewards. **Bold text** indicates the highest score excluding video experts.

Metaworld-50k	<i>BCV-LR (ours)</i>	<i>BCO</i> [26]	<u>DrQv2</u> [3]	Expert Videos
Faucet-open	<b>0.82 ± 0.20</b>	0.13 ± 0.19	0.00 ± 0.00	1.00
Reach	<b>0.63 ± 0.25</b>	0.03 ± 0.05	0.13 ± 0.12	1.00
Drawer-open	<b>0.92 ± 0.12</b>	0.13 ± 0.09	0.00 ± 0.00	1.00
Faucet-close	<b>0.98 ± 0.04</b>	0.00 ± 0.00	0.50 ± 0.28	1.00
Mean Success Rate	<b>0.84</b>	0.07	0.16	1.00

## B.3 Additional Comparison with LAPO’s BC Variant

In Section 4, we use LAPO [21] as a reinforcement learning baseline, which follows its original paper and official implementation. In this section, we replace LAPO’s online RL loss with behavior cloning (BC) loss, obtaining its reward-free BC variant, and then compare it with BCV-LR. The results in Table 7 show that LAPO-BC works well in some tasks, but our BCV-LR still performs better.

Table 7: The comparison between BCV-LR and LAPO’s BC variant. *Italics* indicate using expert videos, and underlines denote using environmental rewards. **Bold text** indicates the highest success rate excluding video experts.

Task	<i>BCV-LR (ours)</i>	<i>LAPO-BC</i> [21]	<u>PPO</u> [63]	Expert Videos
Fruitbot	<b>27.5 ± 1.5</b>	6.2 ± 1.9	-1.9 ± 1.0	29.9
Heist	<b>9.3 ± 0.1</b>	9.2 ± 0.3	3.7 ± 0.2	9.7
Bossfight	<b>10.3 ± 0.3</b>	0.0 ± 0.0	0.1 ± 0.1	11.6
Chaser	<b>3.1 ± 0.5</b>	0.6 ± 0.0	0.4 ± 0.2	10.0

#### B.4 Ablation&Hyper-parameter Sensitivity: Self-supervised Reconstruction Loss

In this section, we further provide the ablation study of the extra reconstruction loss employed in Section 3.1.1. In video games, useful visual information is abundant and scattered. Unlike contrastive learning, which understands the image as a whole, the reconstruction task forces the feature encoder to focus on these key pixels. The results demonstrate that this extra objective can indeed improve the performance, as long as it is not set too large, which is shown in Figure 6.

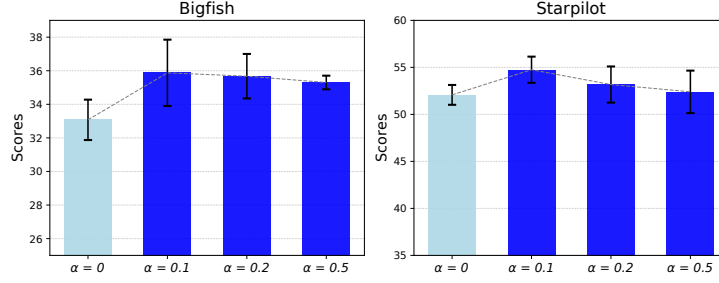


Figure 6: Ablation and hyper-parameter sensitivity analysis of the extra reconstruction loss in self-supervised latent feature pre-training.

#### B.5 Ablation&Hyper-parameter Sensitivity: Finetuning Latent Actions with World Model

In this section, we demonstrate whether finetuning the latent actions online with the world model yields better performance. The results in Figure 7 indicate that constraining the latent action finetuning with the pre-trained world model is helpful. It is consistent with our intuition, because we aim to extract the expert actions for policy cloning, while the online collection is not expert-level in most of the time. The pre-trained world model can provide the expert dynamics-based knowledge to alleviate the effect of the distribution difference.

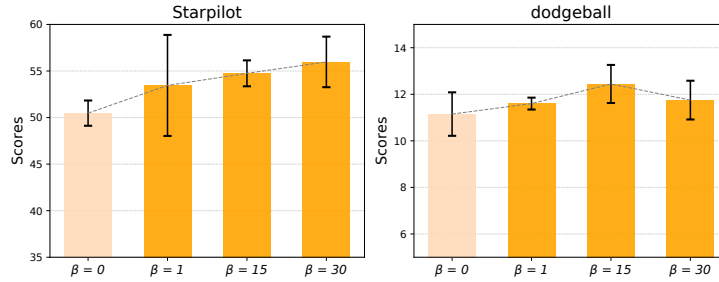


Figure 7: Ablation and hyper-parameter sensitivity analysis of the dynamics-constrained loss in online latent action finetuning.

## B.6 Ablation: Freezing Latent Features

In this section, we provide ablation experiments to demonstrate whether it is necessary to further fine-tune the pre-trained self-supervised latent features with the other objectives in BCV-LR. We finetune the self-supervised encoder with  $\mathcal{L}_{la}$  and  $\mathcal{L}_{ft}$ , respectively. The results in Table 8 demonstrate that whether finetuning self-supervised visual features doesn’t yield an apparent effect on policy performance. This phenomenon has also been observed in self-supervised RL [59, 57], leading some works to fine-tune self-supervised features while others opt to freeze them.

Table 8: Comparison of policy learning performance between freezing and fine-tuning BCV-LR self-supervised latent features.

Task	Finetuned via $\mathcal{L}_{la}$	Finetuned via $\mathcal{L}_{ft}$	Frozen	Expert videos
reacher_hard	876 $\pm$ 15	<b>906 <math>\pm</math> 65</b>	900 $\pm$ 31	967
finger_spin	937 $\pm$ 26	920 $\pm$ 57	<b>942 <math>\pm</math> 48</b>	981

## B.7 Schedule of Environment Interactions

In the online stage of BCV-LR, a) we allow the agent to interact with the environment for a fixed number of times using its policy and collect transitions to enrich the experience buffer. Immediately after that, b) we perform finetuning of the latent action and training of the action decoder on the experience buffer. Then, c) we train the latent policy to imitate the finetuned latent action predicted from expert videos. After this, the BCV-LR policy is improved, and we return to part a) to collect better training data, which forms a cyclic online policy learning. Under the default experimental settings, we set the number of interactions for each cycle at a relatively large value from start to finish (for example, we fixed the number of collected transitions in part a) as 1000 in DMControl). In this section, we try a much smaller interaction number (set to 2) and correspondingly reduce the number of update times for latent actions (set to 1) and latent policies (set to 2) in each cycle. We denote this variant as ‘BCV-LR(1000->2)’. The results in Table 9 show that our approach can still achieve effective policy learning, demonstrating its robustness to the schedule of environment interactions.

Table 9: Comparison of BCV-LR’s performance under different schedules of environment interactions.

Task	BCV-LR (1000->2)	BCV-LR	DrQv2[3]	Expert videos
reacher_hard	875 $\pm$ 65	<b>900 <math>\pm</math> 31</b>	92 $\pm$ 98	967
finger_spin	<b>956 <math>\pm</math> 20</b>	942 $\pm$ 48	374 $\pm$ 264	981

## B.8 Offline Latent Action Decoding

In this section, we let BCV-LR perform latent action finetuning and policy imitation with a few offline action-labeled expert transitions, in a fashion akin to LAPA [19]. Concretely, we maintain the original pre-training stage and use 10k offline action-labeled expert transitions to achieve offline latent action finetuning and policy cloning with the original losses. The results in Table 10 (RL denotes DrQv2 [3] for DMControl tasks and PPO [63] for Procgen tasks) demonstrate that BCV-LR can also achieve offline imitation learning well if expert actions are provided.

Table 10: BCV-LR can also accomplish latent action decoding and fine-tuning using offline action-labeled expert data.

Task	BCV-LR-offline	BCV-LR	RL	Expert videos
reacher_hard	<b>938 <math>\pm</math> 44</b>	900 $\pm$ 31	92 $\pm$ 98	967
finger_spin	<b>978 <math>\pm</math> 7</b>	942 $\pm$ 48	374 $\pm$ 264	981
Fruitbot	<b>27.7 <math>\pm</math> 0.4</b>	27.5 $\pm$ 1.5	-1.9 $\pm$ 1.0	29.9

## B.9 Limitation Analysis

Despite the remarkable results across different kinds of control tasks (including both discrete control and continuous control), BCV-LR faces covariate shift [25], the same issues as all methods based on behavior cloning. This issue restricts the ability to handle sequential decision-making tasks, such as complex robot locomotion. In this section, we provide the comparison between the IRL method LAIFO [50] and our BCV-LR on two continuous tasks: the balance control "reacher\_hard" and the locomotion task "walker\_walk". As shown in Figure 8, when the environmental interactions are limited (only 100k steps), BCV-LR exhibits significant advantages on both tasks.

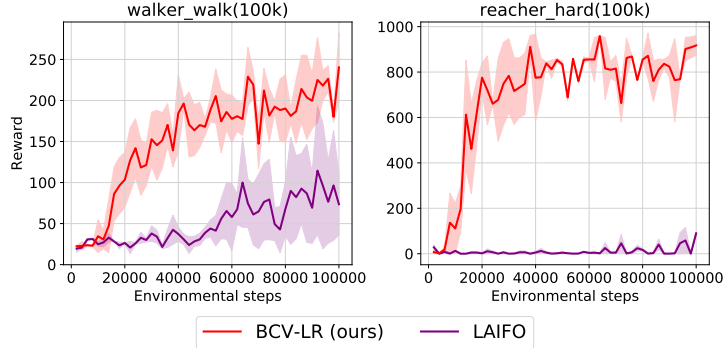


Figure 8: The training curves of BCV-LR and LAIFO when 100k steps are allowed. BCV-LR exhibits advantages on both locomotion and balance control.

However, if enough environmental steps (500k) are permitted, behavior cloning-based BCV-LR’s performance growth stagnates after efficient learning in the sequential decision-making task "walker\_walk", while RL-based LAIFO can continue to improve.

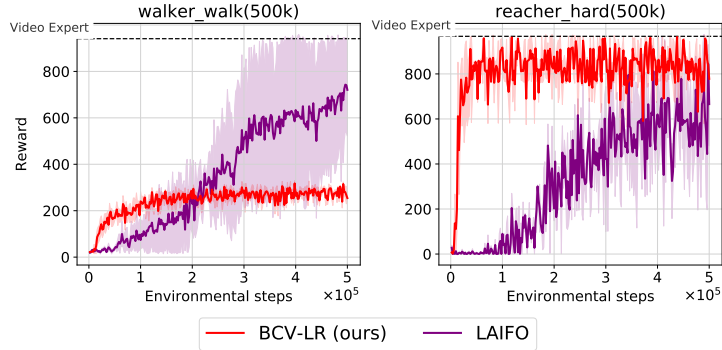


Figure 9: The training curves of BCV-LR and LAIFO when 500k steps are allowed. BCV-LR is limited in sequential decision-making task, which is due to the covariate shift of the behavior cloning.

Some recent works have demonstrated the feasibility of combining inverse RL and behavior cloning to address state-based ILO problems [65]. To this end, integrating BCV-LR with existing inverse rewards or designing inverse rewards based on BCV-LR latent representations are both viable improvement directions, which we leave for future work.

## C Experimental Details

### C.1 Introductions of Environments

In this paper, we employ diverse and challenging visual control tasks to provide a comprehensive evaluation of the proposed BCV-LR, as shown in Figure 10. The top two rows are discrete Progen tasks [30], the third row contains continuous DMControl tasks [31], while the bottom row contains Metaworld manipulation tasks [32].

The first row from left to right is Coinrun, Starpilot, Caveflyer, Dodgeball, Leaper, Maze, Bigfish, and Heist.

The second row from left to right is Fruitbot, Chaser, Miner, Jumper, Climber, Plunder, Ninja, and Bossfight.

The third row from left to right is jaco\_reach\_bottom\_left, point\_mass\_easy, finger\_spin, reacher\_easy, cheetah\_run\_backward, ball\_in\_cup\_catch, jaco\_reach\_top\_right, and reacher\_hard.

The bottom row from left to right is Faucet-open, Reach, Drawer-open, and Faucet-close.

See their paper for the detailed task description.

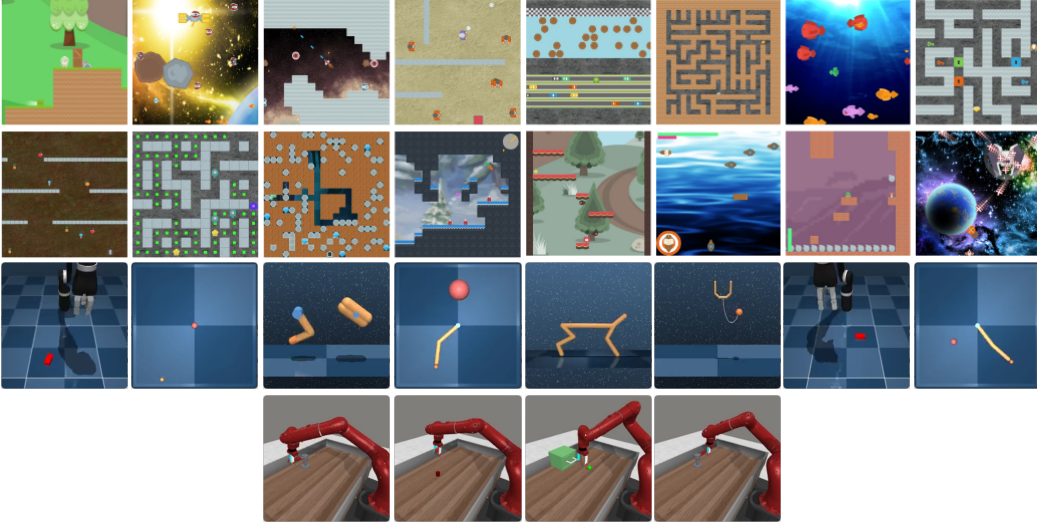


Figure 10: Screenshot of diverse and challenging visual control tasks employed in this paper.



## C.2 Hyper-parameter Settings

Table 11: Default hyper-parameter settings in discrete Procgen. The "lr" denotes "learning rate", while the "ft" denotes "finetuning". There are additionally 60k latent behavior cloning update times before the first round of the online stage to ensure that the latent policy fully imitates the pre-trained latent actions. For Miner, Climber, and Chaser, the "online latent action predictor ft lr" is  $1e-3$  and the  $\beta$  is set to 1. For Plunder, the  $\alpha$  is 1.

Hyper-parameter	Setting
Frame shape	$64 \times 64 \times 3$
Frame stack	1
Action repeat	1
Action type	Discrete
Action dimension	15
Video steps	8M
Video expert training steps	50M
Latent feature pt times	20000
Feature encoder lr	$3e-5$
Random shift padding upper bound	1
Reconstruction coefficient $\alpha$	$1e-1$
EMA update frequency	2
EMA Momentum	0.05
VQ codebook number	2
VQ discrete latent number	4
VQ latent embedding dimension	16
VQ latent embedding number	64
Latent action dimension	128
Latent action pt times	10000
Latent action predictor pt lr	$3e-4$
World model pt lr	$3e-4$
Online steps	100000
Online env numbers	64
Online update frequency	64
Latent action ft update times	60
Latent action decoder lr	$1e-3$
Latent action predictor ft lr	$2e-5$
World model loss coefficient $\beta$	15
World model ft	Frozen
Behavior cloning update times	500
Latent policy lr	$2e-4$

Table 12: Default hyper-parameter settings in DMControl. The "lr" denotes "learning rate", while the "ft" denotes "finetuning". There are additionally 60k latent behavior cloning update times before the first round of the online stage to ensure that the latent policy fully imitates the pre-trained latent actions. For 'point\_mass\_easy', the random shift padding is up to 1. For the domain jaco and reacher, the update time of latent action is set to the fixed value of 100 while the  $\beta$  and the world model finetuning learning rate are both  $1e - 3$ . For Metaworld tasks, they share hyper-parameters with DMControl, except for the action dimension set to 4, the online interaction frames set to 50k, and the latent feature self-supervised task changed to contrastive learning.

Hyper-parameter	Setting
Frame shape	$64 \times 64 \times 3$
Frame stack	3
Action repeat	2
Action type	Continuous
Action dimension	Refer to [31]
Video frames	200k
Video expert training frames	1M
Temperature hyper-parameter $\tau$	0.1
Latent feature pt times	50000
Feature encoder lr	$1e - 4$
Random shift padding upper bound	4
Numbers of doubly-normalization	3
EMA update frequency	2
EMA Momentum	0.05
VQ codebook number	2
VQ dicrete latent number	4
VQ latent embedding dimension	16
VQ latent embedding number	64
Latent action dimension	128
Latent action pt times	50000
Latent action predictor pt lr	$3e - 4$
World model pt lr	$3e - 4$
Online interaction frames	100000
Online env numbers	1
Online update frequency	1000
Latent action ft update epoch	2
Latent action decoder lr	$1e - 3$
Latent action predictor ft lr	$1e - 3$
World model loss coefficient $\beta$	$1e - 5$
World model ft lr	$1e - 5$
Behavior cloning update times	1000
Latent policy lr	$1e - 3$