A Algorithm

Algorithm 1 Automated Auxiliary Loss Search

1: **Initialization:** Randomly generate (bootstrapping) P auxiliary loss functions $\{\mathcal{L}_p\}^P_{\text{stage-1}}$ and P parameterized policy $\{\pi_{\omega_p}\}^P_{\text{stage-1}}$;
2: **for** $i = 1, 2, \dots N$ **do**
3: Optimize policies $\{\pi_{\omega_p}\}^P_{\text{stage-}i}$ with RL loss $\mathcal{L}_{\text{RL}}$ and corresponding Auxiliary loss $\{\mathcal{L}_p\}^P_{\text{stage-}i}$.
4: Evaluate performance (AULC scores) of each RL agent $\{\mathcal{E}_p\}^P_{\text{stage-}i}$ and select top T candidates $\{\mathcal{L}_t\}^T_{\text{stage-}i}$.
5: Apply mutations and loss rejection check (introduced in Section 3.2.2) on top T candidates $\{\mathcal{L}_t\}^T_{\text{stage-}i}$ to generate new auxiliary loss candidates $\{\mathcal{L}_p\}^P_{\text{stage-}(i+1)}$
6: **end for**
7: Cross validate (introduced in Section 4.2) top-performing candidates during evolution to get the optimal auxiliary loss $\mathcal{L}^*$.
8: **return** $\mathcal{L}^*$

B Examples of Loss Functions

We show examples of existing $\mathcal{L}_{\text{RL}}$ and $\mathcal{L}_{\text{Aux}}$ below.

RL loss instances. RL losses are the basic objectives for solving RL problems. For example, when solving discrete control tasks, the Deep Q Networks (DQN) [31] only fit the Q function, where $\mathcal{L}_{\text{RL}}$ is minimizing the error between Q_ω and $Q_{\hat{\omega}}$ (target Q network):

$$\mathcal{L}_{\text{RL}} = \mathcal{L}_{\text{RL},Q}(\omega; \mathcal{E}) = \mathbb{E}_{s,r\sim\mathcal{E},a\sim\pi}(Q_\omega(s_t, a_t) - (r_t + \gamma \max_a Q_{\hat{\omega}}(s_{t+1}, a_t)))^2 \tag{4}$$

However, for continuous action space, the agent is always required to optimize a policy function alongside the Q loss as in eq. (4). For instance, Soft Actor Critic (SAC) [14] additionally optimizes the policy by policy gradient like:

$$\mathcal{L}_{\text{RL}} = \mathcal{L}_{\text{RL},Q} + \mathcal{L}_{\text{RL},\pi}, \quad \mathcal{L}_{\text{RL},\pi}(\omega; \mathcal{E}) = \mathbb{E}_{s,r\sim\mathcal{E},a\sim\pi}(-\min_{i=1,2} Q_{\hat{\omega}_i}(s_t, a_t) + \alpha \log \pi_\omega(a_t|s_t)) \tag{5}$$

Auxiliary loss instances. Besides $\mathcal{L}_{\text{RL}}$, adding an auxiliary loss $\mathcal{L}_{\text{Aux}}$ helps to learn informative state representation for the best learning efficiency and final performance. For example, auxiliary loss of forward dynamics measures the mean squared error of state in the latent space:

$$\mathcal{L}_{\text{Aux}}(\theta; \mathcal{E}) = \|h(g_\theta(s_t), a_t) - g_{\hat{\theta}}(s_{t+1})\|_2, \tag{6}$$

where h denotes a predictor network. Another instance, Contrastive Unsupervised RL (CURL) [23] designs the auxiliary loss by contrasitive similarity relations:

$$\mathcal{L}_{\text{Aux}}(\theta; \mathcal{E}) = \frac{\exp(g_\theta(s'_t)^\top W g_{\hat{\theta}}(s'_{t_+}))}{\exp(g_\theta(s'_t)^\top W g_{\hat{\theta}}(s'_{t_+})) + \sum_{i=0}^{K-1} \exp(g_\theta(s'_t)^\top W g_{\hat{\theta}}(s'_i))}, \tag{7}$$

where s'_t and s'_{t_+} are states of the same state s_t after different random augmentations, and W is a learned parameter matrix.

C Implementation Details

C.1 Architecture

C.1.1 State Encoder Architectures

In Figure 8, we demonstrate the overall architecture when auxiliary loss is used. The architecture is generally identical to architectures adopted in CURL [23]. "Image-based" and "1-layer DenseMLP" are the architectures we used in our experiments. "MLP" and "4-layer DenseMLP" are for ablations. Ablation details are given in Appendix D.6.

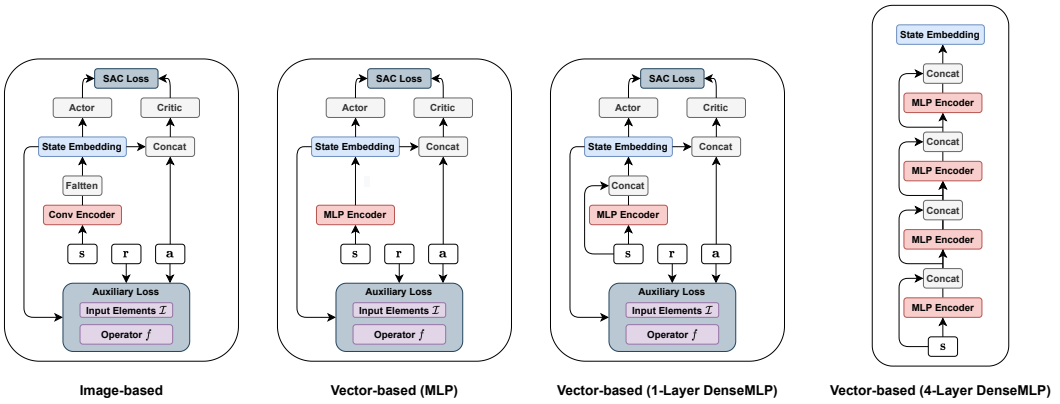


Figure 8: Network structures of image-based RL and vector-based RL with auxiliary losses.

C.1.2 Siamese Network

For a fair comparison with baseline methods, we follow the same Siamese network structure for representation learning as CURL [23]. As shown in Figure 2, when computing targets $\hat{y}$ for auxiliary losses, we map states to state embeddings with a target encoder. We stop gradients from target encoder $\hat{\theta}$ and update $\hat{\theta}$ in the exponential moving averaged (EMA) manner where $\hat{\theta}' = \tau\theta + (1-\tau)\hat{\theta}$. This step, i.e., to freeze the gradients of the target encoder, is necessary when the loss is computed without negative samples. Otherwise, encoders will collapse to generate the same representation for any input. We have verified this in our early experiments.

C.2 Loss Operators

Instance Discrimination Our implementation is based on InfoNCE loss [33]:

$$L = \log \frac{\exp(\phi(y, \hat{y}))}{\exp(\phi(y, \hat{y})) + \sum_{i=0}^{K-1} \exp(\phi(y, y_i))} \tag{8}$$

The instance discrimination loss can be interpreted as a log-loss of a K-way softmax classifier whose label is $\hat{y}$. The difference between discrimination-based loss operators lies in the discrimination objective ϕ used to measure agreement between $(y, \hat{y})$ pairs. Inner employs inner product $\phi(y, \hat{y}) = y^\top \hat{y}$ while Bilinear employs bilinear product $\phi(y, \hat{y}) = yW\hat{y}$, where W is a learnable parameter matrix. Cosine uses cosine distance $\phi(y, \hat{y}) = \frac{y^\top \hat{y}}{\|y\| \cdot \|\hat{y}\|}$ for further matrix calculation. As for losses implemented with cross entropy calculation without negative samples, we only take diagonal elements of matrix M where $M_{i,j} = \phi(y_j, \hat{y}_i)$ for cross entropy calculation.

Mean Squared Error The implementation of MSE-based loss operators are straightforward. MSE loss operator = $(y - \hat{y})^2$ while normalized MSE = $(\frac{y}{\|y\|} - \frac{\hat{y}}{\|\hat{y}\|})^2$. When combined with negative samples, MSE loss operator (with negative pairs) = $(y - \hat{y})^2 - (y - y_i)^2$ while normalized MSE (with negative pairs) = $(\frac{y}{\|y\|} - \frac{\hat{y}}{\|\hat{y}\|})^2 - (\frac{y}{\|y\|} - \frac{y_i}{\|y_i\|})^2$.

C.3 Evolution Strategy

Horizon-changing Mutations There are two kinds of mutations that can change horizon length. One is to decrease horizon length. Specifically, we remove the last time step, i.e., $(s_{t+k}, a_{t+k}, r_{t+k})$ if the target horizon length is k. The other is to increase horizon length, in which we append three randomly generated bits to the given masks at the end. We do not shorten the horizon when it becomes too small (less than 1) or lengthen the horizon when it is too long (exceeding 10).

Mutating Source and Target Masks When mutating a candidate, the mutation on the *source* and the *target* are independent except for horizon change mutation where two masks should either both increase horizon or decrease horizon.

Initialization At each initialization, we randomly generate 75 auxiliary loss functions (every bit of masks are generated from Bernoulli(p) where $p = 0.5$.) and generate 25 auxiliary loss functions with prior probability, which makes the auxiliary loss have some features like forward dynamics prediction or reward prediction. The prior probability for generating the pattern of forward dynamics prediction is: (i) every bit of states from *target* is generated from Bernoulli(p) where $p = 0.2$; (ii) every bit of actions from *source* is generated from Bernoulli(p) where $p = 0.8$; (iii) every bit of states from *target* is generated by flipping the states of *source*; (iv) The other bits are generated from Bernoulli(p) where $p = 0.5$. The prior probability for generating the pattern of reward prediction is: (i) every bit of rewards from *target* is generated from Bernoulli(p) where $p = 0.8$; (ii) Every bit of states and actions from *target* is 0; (iii) The other bits are generated from Bernoulli(p) where $p = 0.5$.

C.4 Training Details

C.4.1 Hyper-parameters in the Image-based Setting

We use the same hyper-parameters for A2LS, SAC-wo-aug, SAC and CURL during the search phase to ensure a fair comparison. When evaluating the searched auxiliary loss, we use a slightly larger setting (e.g., larger batch size) to train RL agents sufficiently. A full list is shown in Table 6.

C.4.2 Hyper-parameters in the Vector-based Setting

We use the same hyper-parameters for A2LS, SAC-Identity, SAC-DenseMLP and CURL-DenseMLP, shown in Table 7. Since training in vector-based environments is substantially faster than in image-based environments, there is no need to balance training cost and agent performance. We use this setting for both the search and final evaluation phases.

C.5 Baselines Implementation

Image-based Setting These following baselines are chosen because they are competitive methods for benchmarking control from pixels. CURL [23] is the main baseline to compare within the image-based setting, which is considered to be the state-of-the-art image-based RL algorithm. CURL learns state representations with a contrastive auxiliary loss. PlaNet [16] and Dreamer [15] are model-based methods that generate synthetic rollouts with a learned world model. SAC+AE [51] uses a reconstruction auxiliary loss of images to boost RL training. SLAC [26] leverages forward dynamics to construct a latent space for RL agents. Note that there are two versions of SLAC with different gradient Updates per agent step: SLACv1 (1:1) and SLACv2(3:1). We adopt SLACv1 for comparison since all methods only make one gradient update per agent step. Image SAC is just vanilla SAC [14] agents with images as inputs.

Vector-based Setting As for the vector-based setting, we compare A2LS with SAC-Identity, SAC and CURL. SAC-Identity is the vanilla vector-based SAC where states are directly fed to actor/critic networks. SAC and CURL use the same architecture of 1-layer densely connected MLP as a state encoder. Note that both A2LS and baseline methods use the same hyper-parameter reported in Table 7 without additional hyper-parameter tuning.

Table 6: Hyper-parameters used in image-based environments.

Hyper-parameter	During Evolution	Final Evaluation of `A2-winner`
Random crop	False for SAC-wo-aug; True for others	True
Observation rendering	(84, 84) for SAC-wo-aug; (100, 100) for others	(100, 100)
Observation downsampling	(84, 84)	(84, 84)
Replay buffer size	100000	100000
Initial steps	1000	1000
Stacked frames	3	3
Actoin repeat	4 (Cheetah-Run, Reacher-Easy) 2 (Walker-Walk);	8 (Cartpole-Swingup); 4 (Others) 2 (Walker-Walk, Finger-Spin)
Hidden units (MLP)	1024	1024
Hidden units (Predictor MLP)	256	256
Evaluation episodes	10	10
Optimizer	Adam	Adam
(β_1, β_2) for actor/critic/encoder	(.9, .999)	(.9, .999)
(β_1, β_2) for entropy α	(.5, .999)	(.5, .999)
Learning rate for actor/critic	1e-3	2e-4 (Cheetah-Run); 1e-3 (Others)
Learning rate for encoder	1e-3	3e-3 (Cheetah-Run, Finger-Spin, Walker-Walk); 1e-3 (Others)
Learning for α	1e-4	1e-4
Batch size for RL loss	128	512
Batch size for auxiliary loss	128	128 (Walker-Walk) 256 (Cheetah-Run, Finger-Spin) 512 (Others);
Auxiliary Loss multipilier λ	1	1
Q function EMA τ	0.01	0.01
Critic target update freq	2	2
Convolutional layers	4	4
Number of filters	32	32
Non-linearity	ReLU	ReLU
Encoder EMA τ	0.05	0.05
Latent dimension	50	50
Discount γ	.99	.99
Initial temperature	0.1	0.1

Table 7: Hyper-parameters used in vector-based environments.

Replay buffer size	100000
Initial steps	1000
Action repeat	4
Hidden units (MLP)	1024
Hidden units (Predictor MLP)	256
Evaluation episodes	10
Optimizer	Adam
(β_1, β_2) for actor/critic/encoder	(.9, .999)
(β_1, β_2) for entropy α	(.5, .999)
Learning rate for actor/critic/encoder	2e-4 (Cheetah-Run); 1e-3 (Others)
Learning for α	1e-4
Batch size	512
Auxiliary Loss multipilier λ	1
Q function EMA τ	0.01
Critic target update freq	2
DenseMLP Layers	1
Non-linearity	ReLU
Encoder EMA τ	0.05
Latent dimension of DenseMLP	40
Discount γ	.99
Initial temperature	0.1

D Additional Experiment Results

D.1 Search Space Pruning

Results of Search Space Pruning Considering that the loss space is huge, an effective optimization strategy is required. Directly grid-searching over the whole space is infeasible because of unacceptable computational costs. Thus some advanced techniques such as space pruning and an elaborate search strategy are necessary. Our search space can be seen as a combination of the space for the input $\mathcal{I}$ and the space for the operator f. Inspired by AutoML works [5, 52] that search for hyper-parameters first and then neural architectures, we approximate the joint search of input and operator in Equation (1) in a two-step manner. The optimal auxiliary loss $\{\mathcal{I}^*, f^*\}$ can be optimized as:

$$\begin{aligned} \max_{\mathcal{L}} \mathcal{R}(M_{\omega^*(\mathcal{L})}; \mathcal{E}) = \max_{\mathcal{I},f} \mathcal{R}(M_{\omega^*(\mathcal{I},f)}; \mathcal{E}) \approx \max_{\mathcal{I}} \mathcal{R}(M_{\omega^*(\mathcal{I},f^*)}; \mathcal{E}) \\ \text{where} \quad f^* \approx \arg\max_{f} \mathbb{E}_{\mathcal{I}}[\mathcal{R}(M_{\omega^*(\mathcal{I},f)}; \mathcal{E})] \end{aligned} \tag{9}$$

To decide the best loss operator, for every f in the *operator* space, we estimate $\mathbb{E}_{\mathcal{I}}[\mathcal{R}(M_{\omega^*(\mathcal{I},f)}; \mathcal{E})]$ with a random sampling strategy. We run 15 trials for each loss operator to estimate performance expectation. For each of 10 possible f in the search space (5 operators with optional negative samples), we run 5 trials on each of the 3 image-based environments (used in evolution) with the same *input elements* $\{s_t, a_t\} \to \{s_{t+1}\}$, as we found that forward dynamics is a reasonable representative of our search space with highly competitive performance. Surprisingly, as summarized in Table 8, the simplest MSE without negative samples outperforms all other loss operators with complex designs. Therefore, this loss operator is chosen for the rest of this paper.

Table 8: Normalized episodic rewards (mean & standard deviation for 5 seeds) of 3 environments used in evolution on image-based DMControl500K with different loss operators.

Loss operator and discrimination	Inner	Bilinear	Cosine	MSE	N-MSE
w/ negative samples	0.979 ± 0.344	0.953 ± 0.329	0.872 ± 0.412	0.124 ± 0.125	0.933 ± 0.360
w/o negative samples	0.669 ± 0.311	0.707 ± 0.299	0.959 ± 0.225	$\mathbf{1.000 \pm 0.223}$	0.993 ± 0.229

Ablation Study on Search Space Pruning As introduced in Appendix D.1, we decompose the full search space into *operator* and *input elements*. Here we try to directly apply the evolution strategy to the whole space without the pruning step. The comparison results are shown in Figure 9. We can see that pruning improves the evolution process, making it easier to find good candidates.

D.2 Learning Curves for A2LS on Image-based DMControl

We benchmark the performance of A2LS to the best-performing image-based baseline (CURL). As shown in Figure 10, the sample efficiency of A2LS outperforms CURL in 10 out of 12 environments. Note that the learning curves of CURL may not match the data in Table 2. This is because we use the data reported in the CURL paper for tabular while we rerun CURL for learning curves plotting, where we find the performance of our rerunning CURL is sightly below the CURL paper.

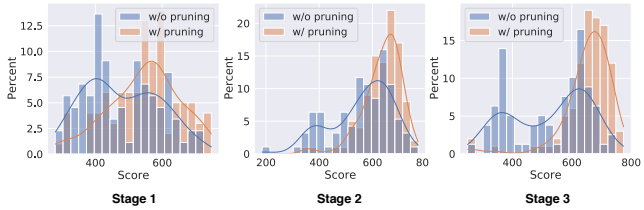


Figure 9: Comparison of evolution with and without pruning by performance histogram.

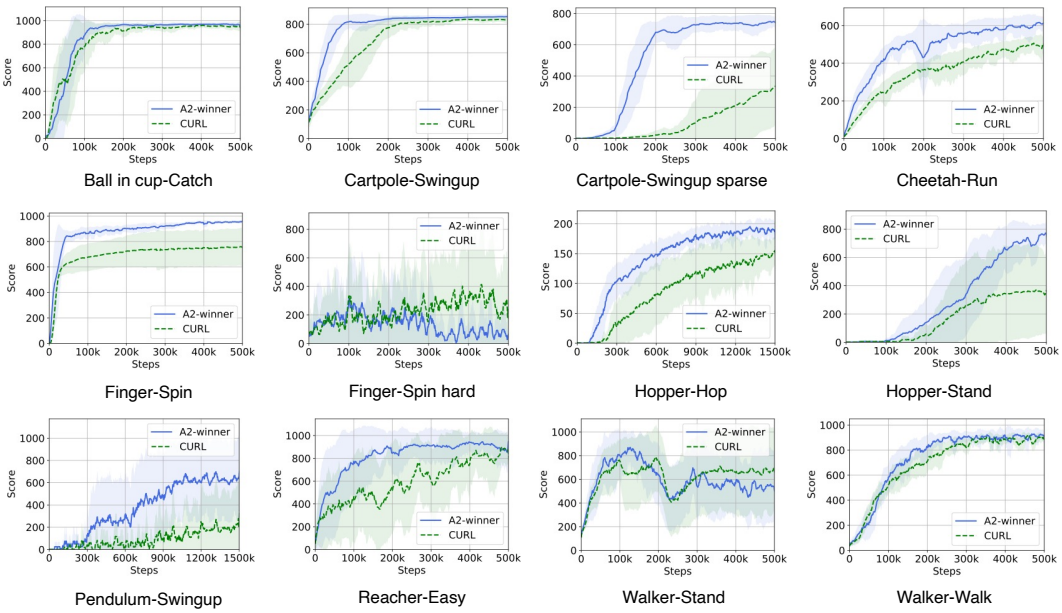


Figure 10: Learning curves of `A2-winner` and CURL on 12 DMC environments. Shadow represents the standard deviation over five random seeds. The curves are uniformly smoothed for visual display. The y-axis represents episodic reward and x-axis represents interaction steps.

D.3 Effectiveness of AULC scores

To illustrate why we use the *area under learning curve* (AULC) instead of other metrics, we select top-10 candidates with different evolution metrics. In practice, AULC is calculated as the sum of scores of all checkpoints during training. Figure 11 demonstrates the usage of AULC score could well balance both sample efficiency and final performance. The learning curves of the top-10 candidates selected by AULC score look better than the other two metrics (that select top candidates simply with 100k step score or 500k step score).

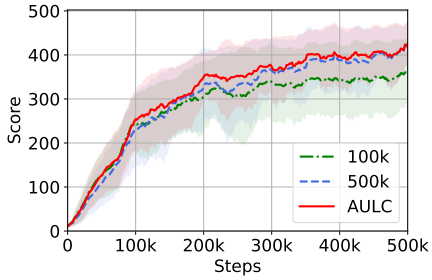


Figure 11: Learning curves of top-10 loss candidates selected with different metrics.

D.4 Comparing Auxiliary Loss with Data Augmentation

Besides auxiliary losses, data augmentation has been shown to be a strong technique for data-efficient RL, especially in image-based environments[25, 22]. RAD [25] can be seen as a version of CURL without contrastive loss but with a better image transformation function for data augmentation. We compare `A2-winner` with RAD in both image-based and vector-based DMControl environments. The learning curves in image-based environments are shown in Figure 12, where no statistically significant difference is observed. As readers may notice, the scores on RAD paper [25] are higher than the RAD and `A2-winner` learning curves reported. To avoid a misleading conclusion that RAD is much stronger than `A2-winner` , we would like to emphasize some key differences between RAD and our implementation: 1) *Large Conv encoder output dim* (RAD: 47, A2LS/CURL: 25);

2) *Larger image size* (RAD: 108, A2LS/CURL: 100); 3) *Larger encoder feature dim* (RAD: 64, A2LS/CURL: 50). We use the hyper-parameters used in CURL for consistency of scores reported in our paper. However, in vector-based environments, as shown in Figure 13, `A2-winner` greatly outperforms RAD. Due to the huge difference between images and proprioceptive features, RAD could not transfer augmentation techniques like random crop and transforms used for images to vectors. Though RAD designs noise and random scaling for proprioceptive features, `A2-winner` shows much better performance on vector-based settings. These results show that recent progress in using data augmentation for RL is still limited to image-based RL while using auxiliary loss functions for RL is able to boost RL across environments with totally different data types of observation. Besides comparing auxiliary losses with data augmentation in DMC, we also provide experimental results in Atari [1]. As shown in Table 3, `A2-winner` significantly outperforms DrQ [22].

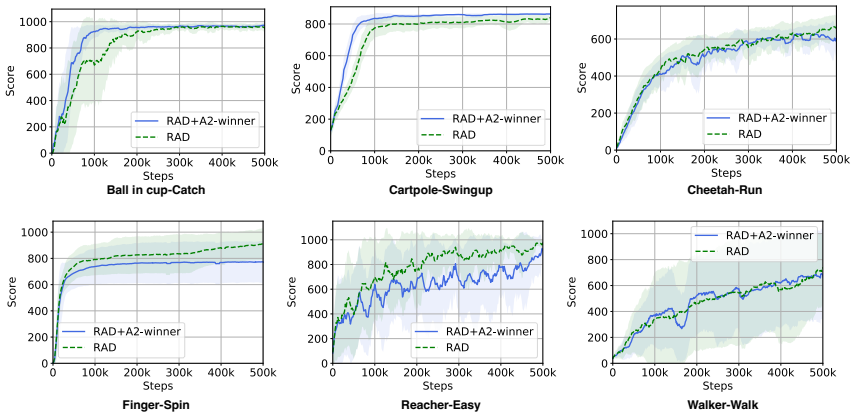


Figure 12: Comparison of learning curves of A2LS and RAD in image-based DMC environments.

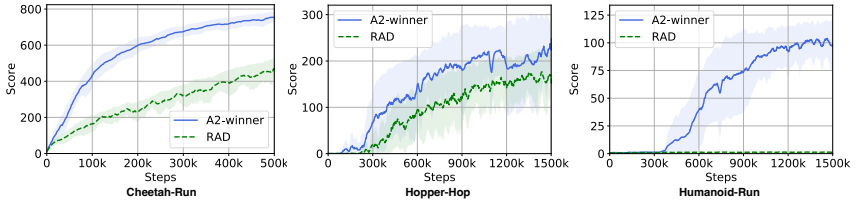


Figure 13: Comparison of learning curves of A2LS and RAD in vector-based DMControl environments.

D.5 Evolution on Vector-based RL

Experiment Settings As for vector-based RL, we use a 1-layer densely connected MLP as the state encoder as shown in Figure 14 due to the low-dimensional state space. So, for this setting, we focus on this simple encoder structure. Additional ablations on state encoder architectures are given in Appendix D.6. In the search phase, we compare A2LS to SAC-Identity, SAC-DenseMLP, CURL-DenseMLP. To ensure a fair comparison, all SAC related hyper-parameters are the same as those reported in the CURL paper. Details can be found in Appendix C.4.2. SAC-Identity is vanilla SAC with no state encoder, while the other three methods (A2LS, SAC-DenseMLP, CURL-DenseMLP) use the same encoder architecture. Different from the image-based setting, there is no data augmentation in the vector-based setting. Note that many environments that are challenging in image-based settings become easy to tackle with vector-based inputs. Therefore we apply our search framework to more challenging environments for vector-based RL, including Cheetah-Run, Hopper-Hop and Quadruped-Run.

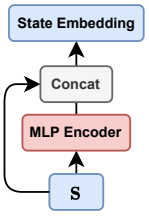


Figure 14: Network architecture of 1-layer DenseMLP state encoder.

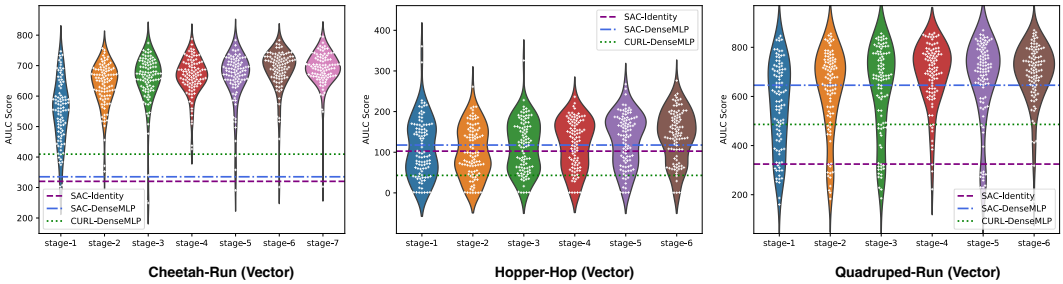


Figure 15: Evolution process in in the three training (vector-based) environments. Every white dot represents a loss candidate, and the score of y-axis shows its corresponding approximated AULC score. The horizontal lines show the scores of baselines. The AULC score is approximated with the average evaluation score at 300k, 600k, 900k, 1200k, 1500k time steps (Cheetah-Run at 100k, 200k, 300k, 400K).

Search Results Similar to image-based settings, we approximate AULC with the average score agents achieved at 300k, 600k, 900k, 1200k, and 1500k time steps[3]. For each environment, we early stop the experiment when the budget of 1,500 GPU hours is exhausted. The evolution process is shown in Figure 15, where we find a large portion of candidates outperform baselines (horizontal dashed lines). The performance improvement is especially significant on Cheetah-Run, where almost all candidates in the population greatly outperform all baselines by the end of the first stage. Similar to image-based settings, we also use cross-validation to select the best loss function, which we call "`A2-winner-v`" here (all the top candidates during evolution are reported in Appendix F)

D.6 Encoder Architecture Ablation for Vector-based RL

Table 9: Normalized episodic rewards of A2LS (mean & standard deviation for 5 seeds of 6 environments) on v DMControl100K with different encoder architectures.

A2LS-MLP (1-layer)	A2LS-MLP (4-layer)	A2LS-DenseMLP (1-layer)	A2LS-DenseMLP (4-layer)
0.919 ± 0.217	0.544 ± 0.360	$\mathbf{1.000 \pm 0.129}$	0.813 ± 0.218

As shown in Figure 14, we choose a 1-layer densely connected MLP as the state encoder for vector-based RL. We conduct an ablation study on different encoder architectures in the vector-based setting. The results are summarized in Table 9, where A2LS with 4-layer encoders consistently perform worse than 1-layer encoders. We also note that dense connection is helpful in the vector-based setting compared with naive MLP encoders.

D.7 Visualization of Loss Landscape

In an effort to reveal why auxiliary losses are helpful to RL, we draw the loss landscape of critic loss of both `A2-winner` and SAC using the technique in [29, 34]. We choose Humanoid-Stand as the testing environment since we observe the most significant advantage of `A2-winner` over SAC on complex robotics tasks like Humanoid. Note that the only difference between `A2-winner` and SAC is whether using auxiliary loss or not. As shown in Figure 16, the critic loss landscape of `A2-winner` appears to be convex during training, while the loss landscape of SAC becomes more non-convex as training proceeds. The auxiliary loss of `A2-winner` is able to efficiently boost Q learning (gaining near 300 reward at 500k steps), while SAC suffers from the poor results of critic learning (gaining near 0 reward even at 1000k time steps). This result shows that such an auxiliary loss might make learning easier from an optimization perspective.

D.8 Histogram of Auxiliary Loss Analysis

The histogram of each pattern analysis is shown in Figure 17.

[3] As for Cheetah-Run, we still use average score agents achieved at 100k, 200k, 300k, 400K and 500k time steps since agents converge close to optimal score within 500k time steps.

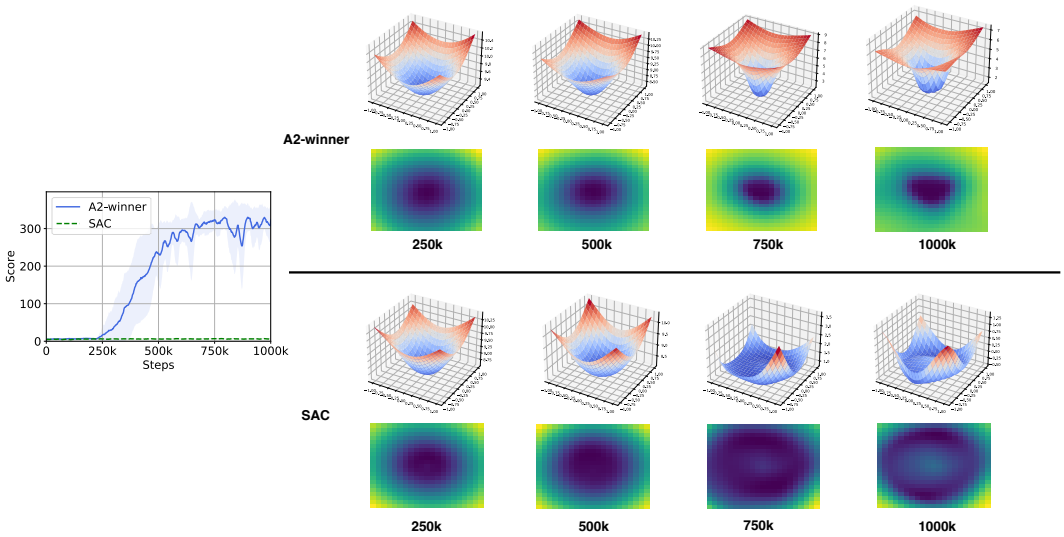


Figure 16: Left: Learning curves of `A2-winner` and SAC on vector-based Humanoid-Stand. Right: Critic Loss Landscape of `A2-winner` (upper right) and SAC (lower right) at 250k, 500k, 750k and 1000k time steps, trained on vector-based Humanoid-Stand. The first row shows 3D surface plots, and the second row shows heatmap plots of loss landscapes.

Table 10: Mean and Median scores (normalized by human score and random score) achieved by A2LS and baselines on 26 Atari games benchmarked at 100k time-steps (Atari100k).

Metric	`A2-winner`	CURL	Eff. Rainbow	DrQ [22]	SimPLe	DER	OTRainbow	SPR	Random	Human
Mean Human-Norm'd	0.568	0.381	0.285	0.357	0.443	0.285	0.264	**0.704**	0.000	1.000
Median Human-Norm'd	0.317	0.175	0.161	0.268	0.144	0.161	0.204	**0.415**	0.000	1.000

D.9 Comparing `A2-winner` with Advanced Human-designed Auxiliary Losses

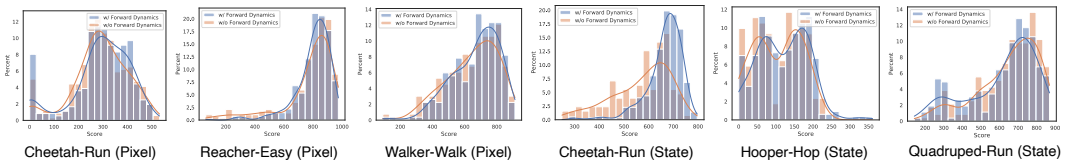
Besides CURL, many recent works (e.g., SPR [39] and ATC [42]) also proposed advanced auxiliary losses that achieve strong performance. Surprisingly, we find that both SPR and ATC designed similar patterns as we conclude in Section 6, like forward dynamics and $n_{\text{target}} > n_{\text{source}}$. Particularly, in ATC, they train the encoder only with ATC loss, and we find the performance of `A2-winner` has better performances than the results reported in their paper: we are $2\times$ more sample efficient to reach 800 scores on Cartpole-Swingup, $2\times$ more sample efficient to reach 100 scores on Hopper-Hop, and $3\times$ more sample efficient to reach 600 scores on Cartpole-Swingup sparse (see Figure 2 of [42]). As for SPR, we find they have superior performance on Atari games benchmark, as shown in Table 10, where `A2-winner` outperform all baselines except SPR. However, note that, our `A2-winner` is only searched on a small set of DMC benchmarks and can still generalize well to discrete-control tasks of Atari, while SPR is designed and only evaluated on Atari environments. In addition, we believe such a gap can arise from different base RL algorithm implementation (A2LS is based on Efficient Rainbow DQN while SPR adopts Categorical DQN) and different hyper-parameters.

D.10 The Trend of Increasing Performance during Evolution

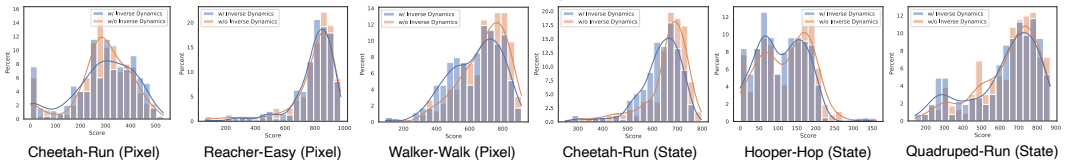
Table 11: Average AULC scores of populations of each stage.

	stage-1	stage-2	stage-3	stage-4	stage-5	stage-6	stage-7	SAC (baseline)
Cheetah-Run	191.75	252.51	258.09	284.53	349.52	351.51	352.57	285.82
Reacher-Easy	674.87	782.75	812.61	823.04	810.15	811.88	827.19	637.60
Walker-Walk	599.38	633.75	716.18	702.49	N/A	N/A	N/A	675.84

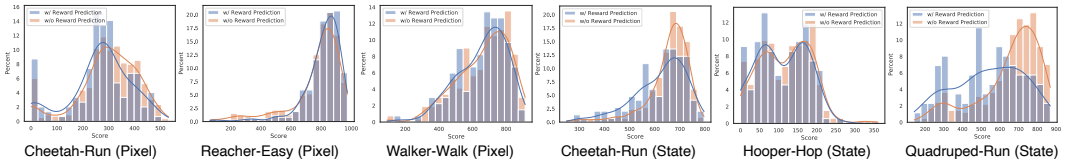
To illustrate the trend of increasing performance during evolution, we provide the average AULC score of populations of each stage in Table 11. As for comparing evoluationary search with random



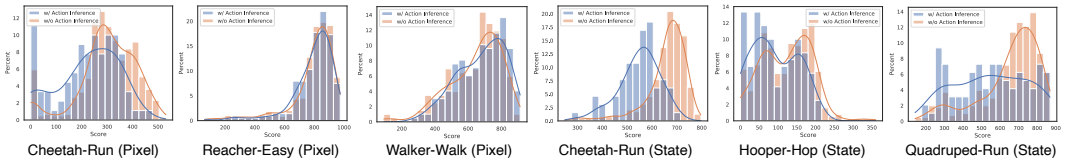
(a) Forward dynamics



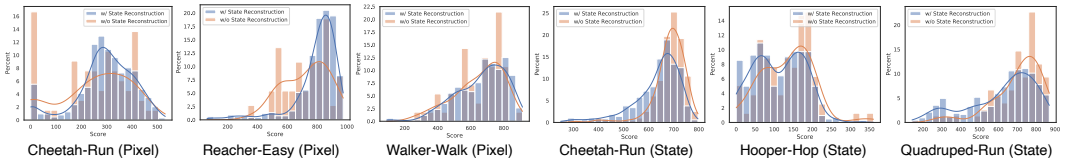
(b) Inverse dynamics



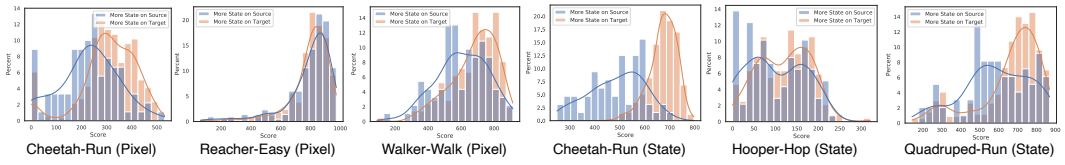
(c) Reward prediction



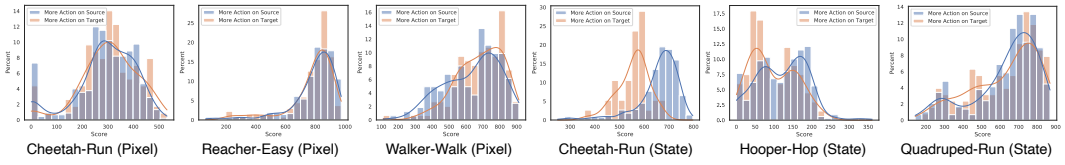
(d) Action inference



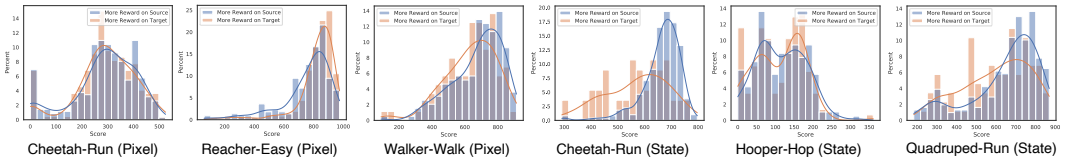
(e) State reconstruction



(f) State source & target



(g) Action source & target



(h) Reward source & target

Figure 17: Histogram of statistical analysis of auxiliary loss candidates in six evolution processes. The x-axis represents approximated AULC score while the y-axis represents the percentage of the corresponding bin of population. Best viewed in color.

Table 12: Average AULC scores of Top-5 candidates of each stage.

	stage-1	stage-2	stage-3	stage-4	stage-5	stage-6	stage-7	SAC (baseline)
Cheetah-Run	398.18	424.27	428.08	485.54	487.94	482.65	498.46	285.82
Reacher-Easy	931.27	950.61	943.83	938.91	954.77	955.02	969.43	637.60
Walker-Walk	834.09	883.77	896.52	880.73	N/A	N/A	N/A	675.84

sampling, we can take the *stage-1 of each evolution procedure* as random sampling. As shown in Table 11, the average performance of the stage-1 population (i.e., random sampling) is even worse than SAC in Cheetah-Run and Walker-Walk. Nevertheless, as evolution continues, the performance of the evolved population in the following stages improves significantly, surpassing the score of SAC.

To illustrate the trend of increasing performance of best individuals during evolution, we provide the average AULC score of the top 5 candidates of the population at each stage in Table 12. As shown in Table 12, there is an obvious trend that the performance of the best individuals in the population at each stage continues to improve and also outperformed the baseline by a large margin during the evolution across all the three training environments.

E Search Space Complexity Analysis

The search space size is calculated by the size of *input element* space multiplying by the size of the loss operator space.

For *input elements*, the search space for input elements is a pair of binary masks $(m, \hat{m})$, each of which is up to length $(3k + 3)$ if the length of an interaction data sequence, i.e., horizon, is limited to k steps. In our case, we set the maximum horizon length $k_{\text{max}} = 10$. we calculate separately for each possible horizon length k. When length is k, the interaction sequence length $(s_t, a_t, r_t, \cdots, s_{t+k})$ has length $(3k + 3)$. For binary mask $\hat{m}$, there are 2^{3k+3} different options. There are also 2^{3k+3} distinct binary mask m to select targets. Therefore, there are 2^{6k+6} combinations when horizon length is fixed to k. As our maximum horizon is 10, we enumerate k from 1 to 10, resulting in $\sum_{i=1}^{10} 2^{6i+6}$.

For *operator*, we can learn intuitively from Table 8 that there are 5 different similarity measures with or without negative samples, resulting in $5 \times 2 = 10$ different loss operators.

In total, the size of the entire space is

$$10 \times \sum_{i=1}^{10} 2^{6i+6} \approx 7.5 \times 10^{20}.$$

F Top-performing Auxiliary Losses

F.1 `A2-winner` and `A2-winner-v`

We introduce all the top-performing auxiliary losses during evolution in this section. Note that MSE is chosen (details are given Appendix D.1) as the loss operator for all the auxliary losses reported below. The source seq_{source} and target seq_{target} of auxiliary loss of `A2-winner` are:

$$\{s_{t+1}, a_{t+1}, a_{t+2}, a_{t+3}\} \rightarrow \{r_t, r_{t+1}, s_{t+2}, s_{t+3}\}, \tag{10}$$

where `A2-winner` is the third-best candidate of stage 4 in Cheetah-Run (Image).

The source seq_{source} and target seq_{target} of auxiliary loss of `A2-winner-v` are:

$$\begin{aligned}\{s_t, a_t, a_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, a_{t+7}, s_{t+8}, a_{t+8}, r_{t+8}\} \\ \rightarrow \{s_{t+1}, s_{t+3}, a_{t+4}, s_{t+6}, s_{t+9}\},\end{aligned} \tag{11}$$

where `A2-winner-v` is the fourth-best candidate of stage 4 in Cheetah-Run (Vector).

These two losses are chosen because they are the best-performing loss functions during cross-validation.

F.2 During Evolution

We report all the top-5 auxiliary loss candidates during evolution in this section.

Table 13: Top-5 candidates of each stage in Cheetah-Run (Image) evolution process

	Cheetah-Run (Image)
Stage-1	$\{r_t, s_{t+1}, a_{t+1}, r_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}\} \rightarrow \{s_t, a_t, s_{t+2}, s_{t+3}, s_{t+4}\}$ $\{s_t, a_t, r_t\} \rightarrow \{s_{t+1}\}$ $\{s_t, a_t, a_{t+1}, r_{t+2}\} \rightarrow \{s_t, a_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, s_{t+3}\}$ $\{s_t, r_t, a_{t+1}, a_{t+2}, a_{t+3}, r_{t+3}, r_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, s_{t+7}\} \rightarrow \{s_t, a_t, s_{t+1}, s_{t+2}, r_{t+2}, r_{t+3}, s_{t+4}, r_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}\}$ $\{s_t, a_t, s_{t+1}, a_{t+1}, s_{t+2}, r_{t+2}\} \rightarrow \{s_t, s_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, s_{t+3}\}$
Stage-2	$\{s_t, a_{t+1}, r_{t+2}, s_{t+4}, r_{t+4}\} \rightarrow \{s_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}, s_{t+5}\}$ $\{s_t, a_t, a_{t+2}, r_{t+2}\} \rightarrow \{s_t, r_t, s_{t+1}, s_{t+2}, r_{t+2}\}$ $\{a_t, r_t, s_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, a_{t+3}, a_{t+4}\} \rightarrow \{s_{t+1}, s_{t+2}, s_{t+3}, a_{t+3}, s_{t+4}\}$ $\{s_t, a_t, r_t, a_{t+1}, r_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, a_{t+4}\} \rightarrow \{s_t, s_{t+1}, s_{t+2}, s_{t+4}, s_{t+5}\}$ $\{r_t, s_{t+1}, r_{t+1}\} \rightarrow \{s_t, a_t, a_{t+1}, s_{t+2}\}$
Stage-3	$\{s_t, a_t, a_{t+2}, r_{t+2}\} \rightarrow \{s_t, s_{t+1}, s_{t+2}, r_{t+2}\}$ $\{s_t, r_t, a_{t+1}, a_{t+3}, r_{t+3}, r_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}\} \rightarrow \{s_t, a_t, s_{t+1}, s_{t+2}, r_{t+2}, r_{t+3}, s_{t+4}, r_{t+5}, s_{t+6}, s_{t+7}, a_{t+7}\}$ $\{s_t, a_t, a_{t+1}, r_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, s_{t+4}, a_{t+4}\} \rightarrow \{s_{t+1}, s_{t+2}, s_{t+4}, s_{t+5}\}$ $\{s_t, a_t, a_{t+1}, r_{t+1}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+4}\} \rightarrow \{s_{t+1}, s_{t+2}, r_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\}$ $\{r_t, s_{t+1}\} \rightarrow \{s_t, a_t, a_{t+1}, s_{t+2}\}$
Stage-4	$\{s_t, s_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, s_{t+4}\} \rightarrow \{a_{t+1}, s_{t+2}, r_{t+2}, s_{t+4}, a_{t+4}, s_{t+5}\}$ $\{s_t, a_t, a_{t+1}, r_{t+1}, r_{t+2}, s_{t+3}, a_{t+3}\} \rightarrow \{s_{t+1}, s_{t+2}, r_{t+3}, s_{t+4}\}$ $\{s_t\} \rightarrow \{s_t, r_t, s_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}\}$ $\{s_t, r_t, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}\} \rightarrow \{s_t, a_t, s_{t+1}, r_{t+1}, r_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\}$ $\{r_t, s_{t+1}, a_{t+1}\} \rightarrow \{s_t, a_t, a_{t+1}, s_{t+2}\}$
Stage-5*	$\{a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}\} \rightarrow \{r_t, s_{t+1}, a_{t+1}, s_{t+2}, s_{t+4}\}$ $\{s_t, a_{t+1}, r_{t+2}, a_{t+3}, s_{t+4}, r_{t+4}\} \rightarrow \{s_{t+1}, s_{t+2}, a_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\}$ †$\{s_{t+1}, a_{t+1}, a_{t+2}, a_{t+3}\} \rightarrow \{r_t, r_{t+1}, s_{t+2}, s_{t+3}\}$ $\{s_t\} \rightarrow \{s_t, r_t, s_{t+1}, r_{t+1}, s_{t+2}\}$ $\{s_t\} \rightarrow \{s_t, r_t, s_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}\}$
Stage-6	$\{a_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, a_{t+3}, a_{t+4}\} \rightarrow \{r_t, s_{t+1}, r_{t+1}, r_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\}$ $\{s_t, a_{t+1}, a_{t+3}, s_{t+4}, r_{t+4}\} \rightarrow \{s_{t+1}, s_{t+2}, a_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\}$ $\{s_t, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}\} \rightarrow \{a_t, s_{t+2}, r_{t+2}, a_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\}$ $\{s_t, a_{t+1}, r_{t+2}, a_{t+3}, s_{t+4}, r_{t+4}\} \rightarrow \{s_{t+1}, s_{t+2}, a_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}, a_{t+5}\}$ $\{s_t, a_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, s_{t+4}, r_{t+4}\} \rightarrow \{s_{t+1}, a_{t+1}, s_{t+2}, r_{t+2}, s_{t+4}, a_{t+4}, r_{t+4}, s_{t+5}\}$
Stage-7	$\{s_t, a_t, r_t, a_{t+1}, r_{t+2}, a_{t+3}, s_{t+4}\} \rightarrow \{a_t, r_t, s_{t+2}, r_{t+3}, s_{t+4}, a_{t+4}, r_{t+4}, s_{t+5}\}$ $\{s_t, r_{t+2}, a_{t+3}, s_{t+4}\} \rightarrow \{a_t, s_{t+1}, s_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, r_{t+4}, s_{t+5}\}$ $\{s_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+4}\} \rightarrow \{r_t, a_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, s_{t+4}, a_{t+4}, s_{t+5}\}$ $\{s_t, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, a_{t+4}\} \rightarrow \{a_t, s_{t+1}, s_{t+2}, r_{t+2}, s_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\}$ $\{s_t, a_{t+1}, r_{t+2}, a_{t+3}, s_{t+4}, r_{t+4}\} \rightarrow \{s_{t+1}, s_{t+2}, r_{t+2}, a_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}, a_{t+5}\}$

∗: Used for cross-validation. †: `A2-winner`.

Table 14: Top-5 candidates of each stage in Reacher-Easy (Image) evolution process

Reacher-Easy (Image)	
Stage-1	$\{s_{t+1}, a_{t+1}\} \rightarrow \{r_t, r_{t+1}\}$
	$\{r_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, a_{t+3}, a_{t+4}, r_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, r_{t+6}, r_{t+7}, s_{t+8}, a_{t+8}, s_{t+9}\} \rightarrow \{a_{t+1}, r_{t+1}, s_{t+2}, s_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, a_{t+5}, a_{t+6}, r_{t+6}, s_{t+7}, a_{t+7}, r_{t+7}, a_{t+8}, r_{t+8}, s_{t+10}\}$
	$\{s_t, r_t, s_{t+1}, a_{t+2}, s_{t+4}, a_{t+4}, r_{t+5}, s_{t+6}, a_{t+6}, a_{t+7}, s_{t+9}, s_{t+10}\} \rightarrow \{s_t, s_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, r_{t+3}, s_{t+4}, a_{t+4}, r_{t+4}, s_{t+5}, s_{t+6}, a_{t+6}, r_{t+7}, r_{t+8}, s_{t+10}\}$
	$\{s_t, r_t, s_{t+1}, a_{t+1}, s_{t+2}, s_{t+3}, a_{t+3}, a_{t+4}, s_{t+5}, r_{t+5}, a_{t+6}, r_{t+6}, s_{t+7}\} \rightarrow \{a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, r_{t+4}, s_{t+6}, a_{t+6}\}$
	$\{s_{t+1}, r_{t+1}, a_{t+2}, r_{t+2}, r_{t+3}, s_{t+4}, a_{t+4}, r_{t+4}, s_{t+5}, s_{t+6}, r_{t+6}\} \rightarrow \{r_t, s_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+4}, s_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}, s_{t+8}\}$
Stage-2	$\{s_t, s_{t+1}, a_{t+1}\} \rightarrow \{r_t, r_{t+1}\}$
	$\{s_t, r_t, a_{t+2}, a_{t+3}, s_{t+4}, a_{t+4}, r_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, a_{t+7}, a_{t+8}\} \rightarrow \{r_t, a_{t+1}, s_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, r_{t+5}, a_{t+6}, s_{t+7}, s_{t+9}\}$
	$\{s_t, a_t, r_t, s_{t+1}, a_{t+2}, a_{t+3}, s_{t+5}\} \rightarrow \{a_t, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, a_{t+4}, r_{t+4}\}$
	$\{s_t, a_t, s_{t+1}, a_{t+3}, s_{t+4}, a_{t+4}, a_{t+5}, s_{t+6}, a_{t+6}, a_{t+7}, r_{t+7}, s_{t+9}, a_{t+9}\} \rightarrow \{s_t, a_t, r_t, s_{t+1}, r_{t+1}, s_{t+2}, r_{t+3}, s_{t+4}, r_{t+4}, s_{t+6}, a_{t+6}, r_{t+7}, a_{t+8}, s_{t+10}\}$
	$\{s_t, a_t, r_t, a_{t+1}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, a_{t+6}, r_{t+6}, a_{t+7}\} \rightarrow \{s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, s_{t+6}, a_{t+6}, s_{t+7}, s_{t+8}\}$
Stage-3	$\{a_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}\} \rightarrow \{r_t, a_{t+1}, r_{t+1}, r_{t+2}\}$
	$\{s_t, a_t, s_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, s_{t+4}, r_{t+4}, a_{t+5}, r_{t+5}, r_{t+6}, a_{t+7}\} \rightarrow \{s_t, a_t, r_t, s_{t+2}, s_{t+3}, a_{t+3}, s_{t+4}, a_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, s_{t+7}, s_{t+8}\}$
	$\{s_t, a_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, s_{t+4}, r_{t+6}, a_{t+7}\} \rightarrow \{r_t, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, a_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}, r_{t+7}, s_{t+8}\}$
	$\{s_t, a_t, s_{t+1}, r_{t+1}, a_{t+2}, a_{t+3}, s_{t+4}, r_{t+4}, s_{t+6}, r_{t+6}, a_{t+7}\} \rightarrow \{s_t, r_t, r_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, s_{t+7}, s_{t+8}\}$
	$\{s_t, a_t, r_t, a_{t+1}, r_{t+1}, a_{t+2}, s_{t+6}, s_{t+7}, a_{t+7}, s_{t+8}\} \rightarrow \{s_t, r_t, s_{t+2}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, s_{t+8}, a_{t+8}\}$
Stage-4	$\{s_t, a_t, r_t, a_{t+1}, a_{t+3}, r_{t+3}, s_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, a_{t+6}, r_{t+6}, a_{t+7}\} \rightarrow \{s_{t+1}, s_{t+2}, s_{t+3}, r_{t+4}, r_{t+5}, r_{t+7}, s_{t+8}\}$
	$\{a_t, s_{t+1}, a_{t+1}, s_{t+2}\} \rightarrow \{r_t, r_{t+1}, a_{t+2}, r_{t+2}\}$
	$\{s_t, a_t, r_t, a_{t+1}, r_{t+3}, s_{t+4}, s_{t+5}, s_{t+6}, r_{t+6}, a_{t+7}\} \rightarrow \{s_{t+1}, s_{t+2}, a_{t+3}, s_{t+4}, r_{t+4}, s_{t+5}, s_{t+6}, s_{t+7}, r_{t+7}, s_{t+8}\}$
	$\{s_t, a_t, r_t, a_{t+2}, a_{t+3}, s_{t+4}, r_{t+4}, r_{t+5}, s_{t+6}, r_{t+6}, a_{t+7}\} \rightarrow \{s_t, r_t, r_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, s_{t+7}, s_{t+8}\}$
	$\{a_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}\} \rightarrow \{r_t, r_{t+1}, r_{t+2}\}$
Stage-5	$\{s_t, a_t, a_{t+1}, a_{t+2}, a_{t+3}, s_{t+4}, s_{t+5}, a_{t+6}, r_{t+6}, a_{t+7}, s_{t+8}\} \rightarrow \{r_t, a_{t+2}, r_{t+2}, a_{t+3}, s_{t+4}, s_{t+5}, a_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}, r_{t+7}, s_{t+8}\}$
	$\{s_t, a_t, s_{t+1}, a_{t+1}, r_{t+1}\} \rightarrow \{r_t, s_{t+2}\}$
	$\{s_t, a_t, r_t, a_{t+1}, r_{t+3}, s_{t+4}, s_{t+5}, s_{t+6}, r_{t+6}, a_{t+7}\} \rightarrow \{s_{t+1}, s_{t+2}, a_{t+3}, s_{t+4}, r_{t+4}, s_{t+5}, r_{t+6}, s_{t+7}, r_{t+7}, s_{t+8}\}$
	$\{s_t, a_t, r_t, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, s_{t+4}, s_{t+5}, a_{t+5}, r_{t+6}, a_{t+7}\} \rightarrow \{s_t, r_t, a_{t+1}, r_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, s_{t+4}, s_{t+6}, a_{t+6}, r_{t+6}, s_{t+7}\}$
	$\{s_t, a_t, s_{t+1}, r_{t+1}, a_{t+2}, s_{t+3}, s_{t+4}, r_{t+6}, a_{t+7}, s_{t+8}\} \rightarrow \{s_t, r_t, r_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, s_{t+4}, a_{t+4}, a_{t+5}, s_{t+6}, r_{t+6}, s_{t+7}, s_{t+8}\}$
Stage-6	$\{s_t, a_t, r_t, a_{t+1}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, r_{t+6}\} \rightarrow \{s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, r_{t+4}, s_{t+6}\}$
	$\{s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, a_{t+4}, r_{t+5}, r_{t+6}, s_{t+7}, s_{t+8}\} \rightarrow \{s_t, r_t, r_{t+2}, a_{t+3}, r_{t+3}, s_{t+4}, r_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, r_{t+6}, s_{t+7}, a_{t+7}, r_{t+7}, s_{t+8}\}$
	$\{s_t, a_t, r_t, r_{t+1}, a_{t+2}, a_{t+3}, s_{t+4}, s_{t+5}, r_{t+5}\} \rightarrow \{s_t, r_t, r_{t+1}, a_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, s_{t+6}, a_{t+6}, r_{t+6}, s_{t+7}, s_{t+8}\}$
	$\{s_t, a_t, s_{t+1}, a_{t+3}, a_{t+4}, a_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}, a_{t+7}, r_{t+7}, s_{t+9}, a_{t+9}\} \rightarrow \{s_t, a_t, r_t, s_{t+1}, r_{t+1}, s_{t+2}, s_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, r_{t+4}, s_{t+6}, a_{t+6}, r_{t+7}, a_{t+8}, a_{t+9}, s_{t+10}\}$
	$\{s_t, a_t, r_t, a_{t+1}, r_{t+1}, a_{t+2}, a_{t+3}, s_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, a_{t+6}, r_{t+6}, a_{t+7}\} \rightarrow \{r_t, s_{t+1}, r_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, s_{t+4}, s_{t+6}, a_{t+6}, r_{t+6}, s_{t+7}, s_{t+8}\}$
Stage-7	$\{s_t, a_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, r_{t+5}, r_{t+6}, a_{t+7}, s_{t+8}\} \rightarrow \{r_t, a_{t+2}, r_{t+2}, r_{t+3}, s_{t+4}, a_{t+4}, a_{t+5}, a_{t+6}, r_{t+6}, s_{t+7}, s_{t+8}\}$
	$\{s_t, a_t, a_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, s_{t+4}, a_{t+5}, r_{t+5}, r_{t+6}, a_{t+7}\} \rightarrow \{s_{t+1}, r_{t+2}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}, r_{t+7}\}$
	$\{s_t, a_t, r_t, a_{t+2}, a_{t+3}, s_{t+4}, s_{t+5}, r_{t+5}, a_{t+6}\} \rightarrow \{s_t, r_t, r_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, r_{t+3}, r_{t+4}, s_{t+6}, s_{t+8}\}$
	$\{s_t, a_t, r_t, a_{t+1}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, r_{t+6}\} \rightarrow \{r_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, a_{t+3}, r_{t+4}, s_{t+6}\}$
	$\{s_t, a_t, r_t, a_{t+1}, r_{t+1}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}, r_{t+5}, s_{t+6}, a_{t+7}, r_{t+7}\} \rightarrow \{s_t, r_t, s_{t+1}, r_{t+1}, a_{t+2}, s_{t+3}, a_{t+3}, r_{t+4}, s_{t+6}, s_{t+8}\}$

Table 15: Top-5 candidates of each stage in Walker-Walk (Image) evolution process

Walker-Walk (Image)	
Stage-1	$\{s_t, a_t, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+4}, a_{t+5}, s_{t+6}, a_{t+6}, a_{t+7}, s_{t+8}, r_{t+8}\} \rightarrow \{s_t, s_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, r_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, r_{t+5}, a_{t+6}, r_{t+6}, r_{t+7}, s_{t+8}, a_{t+8}, s_{t+9}\}$
	$\{s_t, a_t, a_{t+1}, r_{t+1}\} \rightarrow \{a_t, s_{t+1}, r_{t+1}\}$
	$\{r_t, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+3}, s_{t+5}, a_{t+5}, a_{t+6}, r_{t+7}, a_{t+8}\} \rightarrow \{s_t, a_t, s_{t+1}, s_{t+3}, a_{t+3}, s_{t+4}, a_{t+4}, s_{t+6}, s_{t+7}, a_{t+7}, s_{t+8}, s_{t+9}, a_{t+9}, s_{t+10}\}$
	$\{s_t, r_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+4}, r_{t+4}, s_{t+5}, a_{t+5}, a_{t+6}\} \rightarrow \{s_t, a_t, r_t, s_{t+1}, s_{t+2}, s_{t+3}, s_{t+4}, r_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, s_{t+7}\}$
	$\{s_t, r_t, a_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, a_{t+3}, a_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}\} \rightarrow \{s_t, a_t, r_t, a_{t+1}, r_{t+1}, s_{t+2}, s_{t+3}, a_{t+3}, s_{t+5}, a_{t+5}, r_{t+5}, s_{t+6}\}$
Stage-2	$\{s_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+3}, a_{t+3}, a_{t+4}\} \rightarrow \{a_t, r_t, s_{t+1}, s_{t+2}, s_{t+3}, r_{t+3}, a_{t+4}, s_{t+5}\}$
	$\{s_t, r_t, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}, r_{t+3}\} \rightarrow \{a_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, r_{t+3}, s_{t+4}\}$
	$\{r_t, a_{t+1}, s_{t+2}, r_{t+2}, s_{t+3}, r_{t+3}\} \rightarrow \{a_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, s_{t+4}\}$
	$\{s_t, r_t, s_{t+2}, s_{t+3}, r_{t+3}, s_{t+4}\} \rightarrow \{s_t, r_t, s_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\}$
	$\{s_t, r_t, s_{t+1}, s_{t+2}, s_{t+4}, a_{t+4}, a_{t+5}, s_{t+6}, r_{t+6}, s_{t+7}\} \rightarrow \{s_t, r_t, a_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, s_{t+3}, s_{t+4}, a_{t+4}, r_{t+5}, s_{t+6}, a_{t+7}, r_{t+7}, s_{t+8}\}$
Stage-3	$\{s_t, a_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}\} \rightarrow \{s_{t+2}, a_{t+2}, s_{t+3}, r_{t+3}, s_{t+4}\}$
	$\{s_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, a_{t+3}, s_{t+3}\} \rightarrow \{a_t, r_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, r_{t+3}, a_{t+4}, s_{t+5}\}$
	$\{s_t, a_t, a_{t+1}, r_{t+1}\} \rightarrow \{s_{t+2}\}$
	$\{s_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+3}, a_{t+3}, a_{t+4}, r_{t+4}\} \rightarrow \{s_t, a_t, r_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, r_{t+3}, a_{t+4}, s_{t+5}\}$
	$\{s_t, r_t, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}, r_{t+3}\} \rightarrow \{a_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, r_{t+3}, s_{t+4}\}$
Stage-4	$\{s_t, a_t, a_{t+1}\} \rightarrow \{s_{t+1}, a_{t+1}, s_{t+2}\}$
	$\{s_t, r_t, r_{t+1}, s_{t+2}, s_{t+3}, r_{t+3}, r_{t+4}\} \rightarrow \{s_t, a_t, r_t, s_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\}$
	$\{s_t, s_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+4}, a_{t+4}\} \rightarrow \{s_t, a_t, r_t, a_{t+2}, r_{t+2}, a_{t+4}, s_{t+5}\}$
	$\{s_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+3}, s_{t+4}, a_{t+4}\} \rightarrow \{a_t, r_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, r_{t+3}, a_{t+4}, s_{t+5}\}$
	$\{s_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, r_{t+2}, s_{t+3}, a_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}\} \rightarrow \{a_t, r_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, r_{t+3}, a_{t+4}, s_{t+5}\}$

Table 16: Top-5 candidates of each stage in Cheetah-Run (Vector) evolution process

	Cheetah-Run (Raw)
Stage-1	$\{s_t, a_t, r_t, a_{t+1}, r_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$ $\{a_t, r_t, s_{t+2}, a_{t+2}, a_{t+3}, r_{t+3}\} \to \{s_t, s_{t+1}, a_{t+1}, s_{t+3}, s_{t+4}\}$ $\{a_t, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, s_{t+4}, r_{t+4}, a_{t+5}, r_{t+5}, a_{t+7}, r_{t+7}, s_{t+8}, a_{t+8}, r_{t+8}\} \to \{s_t, s_{t+1}, s_{t+3}, a_{t+4}, s_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}, s_{t+9}\}$ $\{a_{t+1}, a_{t+2}, s_{t+3}, a_{t+3}, a_{t+4}, a_{t+5}, r_{t+5}, a_{t+6}, r_{t+7}\} \to \{s_t, a_t, s_{t+1}, s_{t+2}, s_{t+4}, s_{t+5}, s_{t+6}, s_{t+7}, a_{t+7}, s_{t+8}\}$ $\{s_t, a_t, a_{t+1}, a_{t+2}, r_{t+3}, a_{t+4}, r_{t+4}, s_{t+5}, a_{t+5}, a_{t+6}, s_{t+7}, a_{t+7}, s_{t+8}, a_{t+8}, r_{t+8}\} \to \{s_{t+1}, s_{t+2}, s_{t+3}, a_{t+3}, s_{t+4}, s_{t+6}, s_{t+9}\}$
Stage-2	$\{s_t, a_t, r_t, a_{t+1}, r_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, r_t, a_{t+1}, r_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, a_{t+1}, r_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}, r_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, a_{t+6}, a_{t+7}, a_{t+8}, r_{t+8}\} \to \{a_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, s_{t+4}, s_{t+6}, s_{t+9}\}$ $\{s_t, a_t, a_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, a_{t+7}, s_{t+8}, a_{t+8}, r_{t+8}\} \to \{s_{t+1}, s_{t+3}, a_{t+4}, s_{t+6}, s_{t+9}\}$ $\{s_t, a_t, r_t\} \to \{r_t, s_{t+1}\}$
Stage-3	$\{s_t, a_t, r_t, a_{t+1}, r_{t+1}\} \to \{s_{t+1}\}$ $\{s_t, a_t\} \to \{r_t, s_{t+1}\}$ $\{s_t, a_t, r_t, a_{t+1}\} \to \{s_{t+2}\}$ $\{s_t, a_t, r_t\} \to \{s_{t+1}\}$ $\{s_t, a_t, a_{t+1}, r_{t+1}\} \to \{s_t, s_{t+1}, s_{t+2}\}$
Stage-4*	$\{s_t, a_t, r_t, a_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, a_{t+1}, r_{t+1}, a_{t+2}, r_{t+2}, r_{t+3}, r_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, a_{t+6}, a_{t+7}, a_{t+8}, r_{t+8}\} \to \{a_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, s_{t+4}, s_{t+6}, s_{t+9}\}$ $\{s_t, a_t, a_{t+1}, a_{t+2}, r_{t+2}, r_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, r_{t+5}, a_{t+6}, a_{t+7}, a_{t+8}, r_{t+8}\} \to \{a_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, s_{t+4}, a_{t+6}, a_{t+8}, s_{t+9}\}$ †$\{s_t, a_t, a_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, a_{t+7}, s_{t+8}, a_{t+8}, r_{t+8}\} \to \{s_{t+1}, s_{t+3}, a_{t+4}, s_{t+6}, s_{t+9}\}$ $\{s_t, a_t, a_{t+1}, a_{t+2}, r_{t+3}, a_{t+4}, r_{t+4}, s_{t+5}, a_{t+5}, s_{t+7}, a_{t+7}, s_{t+8}, a_{t+8}, r_{t+8}\} \to \{s_{t+1}, s_{t+2}, s_{t+3}, a_{t+3}, s_{t+4}, s_{t+6}, a_{t+8}, r_{t+8}, s_{t+9}\}$
Stage-5*	$\{s_t, a_t, r_t, a_{t+1}\} \to \{s_{t+1}, r_{t+1}\}$ $\{s_t, a_t, a_{t+1}, r_{t+1}\} \to \{s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}\}$ $\{s_t, a_t, r_t, a_{t+1}, r_{t+1}\} \to \{s_{t+1}\}$ $\{s_t, a_t, a_{t+1}, a_{t+2}, r_{t+2}, r_{t+3}, a_{t+4}, r_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, a_{t+6}, a_{t+7}, a_{t+8}, r_{t+8}\} \to \{s_{t+1}, s_{t+2}, s_{t+3}, s_{t+4}, s_{t+6}, a_{t+8}, s_{t+9}\}$ $\{s_t, a_t, r_t, a_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$
Stage-6	$\{s_t, a_t, a_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, r_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, a_{t+6}, a_{t+7}, a_{t+8}, r_{t+8}\} \to \{s_t, a_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, a_{t+3}, a_{t+4}, s_{t+5}, s_{t+6}, a_{t+8}, s_{t+9}\}$ $\{s_t, a_t, a_{t+1}\} \to \{r_t, s_{t+1}, r_{t+1}, s_{t+2}\}$ $\{s_t, a_t, a_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}, r_{t+4}, s_{t+5}, a_{t+5}, a_{t+6}, a_{t+7}, s_{t+8}, a_{t+8}, r_{t+8}\} \to \{a_t, s_{t+1}, s_{t+2}, s_{t+3}, s_{t+6}, a_{t+8}, r_{t+8}, s_{t+9}\}$ $\{s_t, a_t, r_t, a_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, r_t, a_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$
Stage-7	$\{s_t, a_t, r_t, a_{t+1}, r_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, r_t, a_{t+1}, r_{t+1}\} \to \{s_{t+1}, r_{t+1}, s_{t+2}\}$ $\{s_t, a_t, r_t, a_{t+1}\} \to \{r_t, s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, a_{t+1}\} \to \{s_{t+1}, a_{t+1}, s_{t+2}\}$ $\{s_t, a_t, a_{t+1}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}, r_{t+4}, s_{t+5}, a_{t+5}, a_{t+6}, a_{t+7}, a_{t+8}, r_{t+8}\} \to \{s_{t+1}, s_{t+2}, s_{t+3}, a_{t+3}, a_{t+4}, s_{t+6}, s_{t+8}, s_{t+9}\}$

∗: Used for cross-validation. †: `A2-winner-v`.

Table 17: Top-5 candidates of each stage in Hopper-Hop (Vector) evolution process

	Hopper-Hop (Raw)
Stage-1	$\{s_t, a_t\} \to \{r_t, s_{t+1}\}$ $\{a_t, r_t, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+5}, a_{t+5}, r_{t+5}, a_{t+6}, a_{t+7}, r_{t+7}, a_{t+8}, r_{t+8}\} \to \{s_t, s_{t+1}, a_{t+1}, s_{t+4}, a_{t+4}, s_{t+6}, s_{t+7}, s_{t+8}, s_{t+9}\}$ $\{s_t, a_t, s_{t+2}, a_{t+3}, r_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, r_{t+7}, r_{t+8}\} \to \{s_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, s_{t+3}, s_{t+4}, a_{t+6}, s_{t+7}, a_{t+7}, r_{t+7}, a_{t+8}, s_{t+9}\}$ $\{s_t, a_t, s_{t+2}, a_{t+3}, r_{t+3}, a_{t+5}\} \to \{s_t, a_{t+1}, s_{t+2}, s_{t+3}, r_{t+3}, s_{t+4}, r_{t+4}, r_{t+5}, s_{t+6}\}$ $\{s_t, r_t\} \to \{s_t, r_t, s_{t+1}\}$
Stage-2	$\{s_t, a_t, s_{t+1}, a_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, r_t, s_{t+2}, r_{t+2}\} \to \{r_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}\}$ $\{s_t, a_t, s_{t+1}, a_{t+1}, a_{t+4}, s_{t+5}, a_{t+5}, s_{t+6}, a_{t+6}\} \to \{s_{t+2}, r_{t+2}, r_{t+3}, r_{t+4}, s_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, r_{t+6}\}$ $\{r_t, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, a_{t+3}, a_{t+4}, r_{t+4}\} \to \{s_t, a_t, r_t, s_{t+1}, s_{t+2}, s_{t+3}, s_{t+4}, r_{t+4}\}$ $\{s_t, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}\} \to \{s_{t+1}, r_{t+1}, a_{t+2}, s_{t+3}\}$
Stage-3	$\{s_t, a_t, s_{t+1}, a_{t+1}\} \to \{s_t, s_{t+2}\}$ $\{s_t\} \to \{s_t, a_t, r_t, s_{t+1}\}$ $\{s_t, a_t, r_t, s_{t+2}, a_{t+2}, r_{t+2}\} \to \{s_t, s_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}\}$ $\{s_t, a_t, a_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, a_{t+1}\} \to \{s_t, s_{t+1}, s_{t+2}\}$
Stage-4	$\{s_t, r_t, s_{t+2}, a_{t+2}, r_{t+2}\} \to \{r_{t+1}, a_{t+2}, s_{t+3}\}$ $\{s_t, a_t, s_{t+1}, a_{t+1}\} \to \{s_{t+2}\}$ $\{s_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}\} \to \{s_t, r_{t+1}, a_{t+2}, s_{t+3}\}$ $\{s_t, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}\} \to \{s_{t+1}, r_{t+1}, a_{t+2}, s_{t+3}\}$ $\{s_t, a_t, s_{t+1}, a_{t+1}\} \to \{s_{t+1}, s_{t+2}\}$
Stage-5	$\{s_t, a_t, r_t, a_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}\} \to \{s_t, r_{t+1}, a_{t+2}, s_{t+3}\}$ $\{s_t, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}\} \to \{s_t, r_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}\}$ $\{s_t, a_t, r_{t+1}\} \to \{s_t, s_{t+1}, s_{t+2}\}$ $\{s_t, r_t, a_{t+1}, s_{t+2}, r_{t+2}\} \to \{s_{t+1}, a_{t+2}, s_{t+3}\}$ $\{s_t, a_t, s_{t+1}, a_{t+1}\} \to \{r_{t+1}, s_{t+2}\}$
Stage-6	$\{s_t, a_t, a_{t+1}\} \to \{s_t, s_{t+1}, s_{t+2}\}$ $\{s_t, r_t, s_{t+2}, a_{t+2}, r_{t+2}\} \to \{s_t, s_{t+1}, s_{t+2}, s_{t+3}\}$ $\{s_t, a_t, a_{t+1}\} \to \{s_t, s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}\} \to \{r_t, s_{t+1}, s_{t+2}\}$ $\{s_t, a_t, r_t, a_{t+1}, s_{t+2}\} \to \{s_t, s_{t+1}, r_{t+1}\}$

Table 18: Top-5 candidates of each stage in Quadruped-Run (Vector) evolution process

	Quadruped-Run (Raw)
Stage-1	$\{a_t, r_t, s_{t+1}, s_{t+2}, a_{t+2}\} \rightarrow \{s_t, a_{t+1}, s_{t+3}\}$ $\{r_t, s_{t+1}, s_{t+3}, r_{t+3}\} \rightarrow \{s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, s_{t+4}\}$ $\{a_t, a_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}\} \rightarrow \{s_t, s_{t+1}, a_{t+2}, s_{t+4}\}$ $\{s_t, a_t, r_{t+1}, a_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+4}, s_{t+5}\} \rightarrow \{a_t, a_{t+1}, r_{t+1}, a_{t+2}, r_{t+3}, a_{t+4}\}$ $\{s_t, a_t, r_t, s_{t+1}, a_{t+1}, s_{t+3}\} \rightarrow \{r_{t+1}, r_{t+2}\}$
Stage-2	$\{a_t, r_t, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, a_{t+4}\} \rightarrow \{s_t, s_{t+1}, a_{t+1}, s_{t+2}, s_{t+4}, s_{t+5}\}$ $\{s_t, a_t, a_{t+1}, r_{t+1}, r_{t+2}, a_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, r_{t+5}, s_{t+6}, r_{t+6}, a_{t+7}, a_{t+8}, r_{t+8}, s_{t+9}\} \rightarrow \{s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, s_{t+4}, s_{t+5}, s_{t+7}, s_{t+8}\}$ $\{a_{t+1}, r_{t+1}, s_{t+2}, a_{t+3}, r_{t+3}\} \rightarrow \{s_t, a_t, r_t, a_{t+1}, a_{t+3}, s_{t+4}\}$ $\{a_t, a_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, a_{t+4}, a_{t+5}, r_{t+5}, a_{t+6}, r_{t+6}, a_{t+7}, s_{t+8}, a_{t+8}\} \rightarrow \{s_t, s_{t+1}, s_{t+3}, s_{t+4}, s_{t+5}, s_{t+6}, s_{t+7}, s_{t+8}, s_{t+9}\}$ $\{s_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, s_{t+4}\} \rightarrow \{s_t, a_t, s_{t+1}, s_{t+2}, a_{t+2}\}$
Stage-3	$\{a_t, a_{t+1}, a_{t+3}, r_{t+3}, r_{t+4}, a_{t+5}, a_{t+7}, r_{t+7}, s_{t+8}, a_{t+8}\} \rightarrow \{s_t, s_{t+1}, s_{t+2}, s_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}, a_{t+5}, r_{t+5}, s_{t+6}, a_{t+6}, r_{t+6}, s_{t+7}, s_{t+9}\}$ $\{a_t, a_{t+1}, a_{t+3}, r_{t+3}, a_{t+5}, a_{t+7}, s_{t+8}, a_{t+8}\} \rightarrow \{s_t, a_t, s_{t+1}, a_{t+2}, s_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}, s_{t+9}\}$ $\{a_t, r_t, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, r_{t+4}\} \rightarrow \{s_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, s_{t+4}, s_{t+5}\}$ $\{a_t, a_{t+1}, r_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, a_{t+7}, r_{t+7}, s_{t+8}\} \rightarrow \{s_t, r_t, s_{t+1}, s_{t+3}, s_{t+4}, s_{t+5}, s_{t+6}, a_{t+6}, s_{t+7}, s_{t+8}, s_{t+9}\}$ $\{s_t, a_t, r_t, r_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+4}, s_{t+5}\} \rightarrow \{a_t, a_{t+1}, r_{t+1}, a_{t+2}, r_{t+3}\}$
Stage-4	$\{r_t, r_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+4}, s_{t+5}\} \rightarrow \{a_{t+2}, r_{t+3}, a_{t+4}\}$ $\{s_t, a_t, r_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+4}, s_{t+5}\} \rightarrow \{a_t, a_{t+1}, a_{t+2}, r_{t+3}, a_{t+4}\}$ $\{a_t, a_{t+1}, a_{t+3}, r_{t+3}, s_{t+4}\} \rightarrow \{s_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}\}$ $\{s_t, a_t, r_t, r_{t+1}, a_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+4}, s_{t+5}\} \rightarrow \{a_t, a_{t+2}, r_{t+3}, a_{t+4}\}$ $\{s_{t+2}, a_{t+2}, a_{t+3}\} \rightarrow \{s_t, a_t, a_{t+2}, s_{t+3}, a_{t+3}, s_{t+4}\}$
Stage-5	$\{a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, r_{t+2}, a_{t+3}, r_{t+3}\} \rightarrow \{s_t, a_t, r_t, a_{t+1}, r_{t+1}, a_{t+2}, s_{t+3}, a_{t+3}\}$ $\{a_t, r_t, r_{t+1}, a_{t+2}, r_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, r_{t+4}\} \rightarrow \{s_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, a_{t+3}, s_{t+4}, s_{t+5}\}$ $\{a_t, a_{t+1}, a_{t+3}, r_{t+3}, a_{t+4}, r_{t+4}, a_{t+5}, a_{t+7}, r_{t+7}, s_{t+8}, a_{t+8}\} \rightarrow \{s_t, r_t, s_{t+1}, s_{t+2}, s_{t+3}, s_{t+4}, s_{t+5}, a_{t+5}, s_{t+6}, a_{t+6}, r_{t+6}, s_{t+7}, s_{t+8}\}$ $\{a_t, a_{t+1}, s_{t+2}, a_{t+3}, r_{t+3}, a_{t+4}, a_{t+5}, s_{t+6}, a_{t+7}, s_{t+8}, a_{t+8}\} \rightarrow \{s_t, s_{t+1}, s_{t+2}, a_{t+2}, s_{t+3}, s_{t+4}, a_{t+4}, s_{t+5}, s_{t+6}, s_{t+7}, r_{t+7}, r_{t+8}\}$ $\{s_t, a_t, r_t, r_{t+1}, a_{t+2}, s_{t+3}, a_{t+3}, r_{t+3}, s_{t+5}\} \rightarrow \{a_t, a_{t+2}, r_{t+3}, a_{t+4}\}$