APPENDIX

A MISSING PROOFS

Lemma A.1. Assume that $h : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{Y}|}$ can be written as $h(\xi) = f(< \xi, \mathbf{s} >)$, for some $\mathbf{s} \in \mathbb{R}^{|\mathcal{S}|}$, and $f : \mathbb{R}^{|\mathcal{Y}|} \to \mathbb{R}^{|\mathcal{Y}|}$ with parameter ξ. Then, convexity of f implies the convexity of h.

Proof. Let $\xi_1, \xi_2 \in \mathbb{R}^{|\mathcal{S}|}$ and $\tau \in [0, 1]$. We have

$$\begin{aligned} &h(\tau\xi_1 + (1-\tau)\xi_2) = f(< \tau\xi_1 + (1-\tau)\xi_2, \mathbf{s} >) \\ &= f(< \tau\xi_1, \mathbf{s} > + < (1-\tau)\xi_2, \mathbf{s} >) = f(\tau < \xi_1, \mathbf{s} > + (1-\tau) < \xi_2, \mathbf{s} >) \\ &\leq \tau f(< \xi_1, \mathbf{s} >) + (1-\tau) f(< \xi_2, \mathbf{s} >) = \tau h(\xi_1) + (1-\tau) h(\xi_2) \end{aligned} \tag{12}$$

where the last inequality follows from the convexity of f. □

Lemma A.2. Given the updating dynamics: $\bar{\xi}^t = (1-\tau_m)\bar{\xi}^{t-1} + \tau_m \xi^t$. By Lemma 4.1, $f_\xi = f_{\bar{\xi}}$ holds after convergence. As a result, the problem of $\min \mathbb{E}_{\mathbf{x}}[\|f_\xi \circ g_\xi(\mathbf{x}') - f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x})\|]$ is equivalent to the problem of $\min \mathbb{E}_{\mathbf{x}}[\|g_\xi(\mathbf{x}') - g_{\bar{\xi}}(\mathbf{x})\|]$.

Proof. Through Lemma A.1, we know the convexity of a designed function f can give rise to the convexity of the function h with the parameters as the input. Therefore, we design our projections f_ξ and $f_{\bar{\xi}}$ as $f(< \xi, \mathbf{s} >)$ and $f(< \bar{\xi}, \mathbf{s} >)$ respectively. For instance, ReLU and MLP can be adopted here. Using the dynamic: $\bar{\xi}^t = (1-\tau_m)\bar{\xi}^{t-1} + \tau_m \xi^t$ together with $h(\xi)$ mentioned in Lemma A.1, we obtain the divergence of hidden representations $f_\xi \circ g_\xi(\mathbf{x}')$ and $f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x})$,

$$\begin{aligned} \mathbb{E}_{\mathbf{x}}[\|f_\xi \circ g_\xi(\mathbf{x}') - f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x})\|] &= \mathbb{E}_{\mathbf{x}}[\|f_\xi \circ (g_\xi(\mathbf{x}') - g_{\bar{\xi}}(\mathbf{x}))\| \\ &\leq \mathbb{E}_{\mathbf{x}}[\|\xi_f\| \|(g_\xi(\mathbf{x}') - g_{\bar{\xi}}(\mathbf{x}))\|] \\ &= \mathbb{E}_{\mathbf{x}}[\|(g_\xi(\mathbf{x}') - g_{\bar{\xi}}(\mathbf{x}))\|] \|\xi_f\| \end{aligned} \tag{13}$$

where $\|\xi_f\|$ is the parameter of the projection $\|f_\xi\|$. The first equality is determined by the approximation of convergence analysis, which is $f_\xi = f_{\bar{\xi}}$. We use Cauchy–Schwartz inequality here. Note that a premise in this lemma is that the momentum updating reached convergence, which means $f_\xi = f_{\bar{\xi}}$. Minimizing the right side bound equals optimizing the problem of the left side in Eq.(13). When the norm of $\|\xi_f\|$ is fixed, the proof completes. □

Assumption A.1. (L_f-Lipschitzness). Let the projection function $f(\mathbf{s}) : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{Y}|}$ is L_f-Lipschitz, we have that $\forall \mathbf{s}, \mathbf{s}'$, $|f(\mathbf{s}') - f(\mathbf{s})| \leq L_f \|\mathbf{s}' - \mathbf{s}\|$.

Assumption A.2. (L_g-Lipschitzness). Let the projection function $g(\mathbf{x}) : \mathbb{R}^{|\mathcal{X}|} \to \mathbb{R}^{|\mathcal{S}|}$ is L_g-Lipschitz, we have that $\forall \mathbf{x}, \mathbf{x}'$, $|g(\mathbf{x}') - g(\mathbf{x})| \leq L_g \|\mathbf{x}' - \mathbf{x}\|$.

Theorem A.1. (CoIT.) Suppose that Lipschitzness holds for functions g_ξ, $g_{\bar{\xi}}$, f_ξ and $f_{\bar{\xi}}$, respectively. The updating dynamics is: $\bar{\xi}^t = (1-\tau_m)\bar{\xi}^{t-1} + \tau_m \xi^t, \tau_m \in [0, 1]$. For any input $\mathbf{x} \sim \hat{\mathcal{D}}$ and shifted $\mathbf{x}'$ obtained via the transform operator $\nu(\mathbf{x}, \cdot)$, optimizing the conditional divergence in Definition 4.1 means to minimize the upper bound as follows,

$$\mathbb{E}_{\mathbf{x}}\left[d(f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x}), f_\xi \circ g_\xi(\mathbf{x}'))\right] \leq \rho \mathbb{E}_{\mathbf{x}}\left[\|\mathbf{x} - \mathbf{x}'\|\right] \tag{14}$$

where $\rho = L_g\left(CL_f + \|\xi_f\|\right), C = \frac{1+\tau}{1-\tau}, \tau = 1 - \tau_m$ are constants. L_f and L_g are Lipschitz constants of the functions $f(\mathbf{s})$ and $g(\mathbf{x})$ respectively.

Proof. By incorporating Lemma A.2 for the problem of $\min \mathbb{E}_{\mathbf{x}}[\|f_\xi \circ g_\xi(\mathbf{x}') - f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x})\|]$, it leads to a divergence with projections where a triangular inequality holds,

$$d(f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x}), f_\xi \circ g_\xi(\mathbf{x}')) \leq \underbrace{d(f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x}), f_\xi \circ g_\xi(\mathbf{x}))}_{\text{state } \epsilon\text{-approximation}} + \underbrace{d(f_\xi \circ g_\xi(\mathbf{x}), f_\xi \circ g_\xi(\mathbf{x}'))}_{L\text{-Lipschitzness}} \tag{15}$$

Set $\mathbf{s}' = g_\xi(\mathbf{x})$ and $\mathbf{s} = g_{\bar{\xi}}(\mathbf{x})$. Now we use Lemma A.1 and the updating dynamics for the designed projection f_ξ and $f_{\bar{\xi}}$, $\tau_m \in [0, 1]$, we can obtain,

$$h(\bar{\xi}^t) = h((1-\tau_m)\bar{\xi}^{t-1} + \tau_m \xi^t) \leq (1-\tau_m) h(\bar{\xi}^{t-1}) + \tau_m h(\xi^t) \tag{16}$$

By designing a projection that satisfies $h(\bar{\xi}^t) = f^t_{\bar{\xi}}(\mathbf{s})$ and $h(\xi^t) = f^t_{\xi}(\mathbf{s})$, we have

$$f^t_{\bar{\xi}}(\mathbf{s}) \leq (1-\tau_m) f^{t-1}_{\bar{\xi}}(\mathbf{s}) + \tau_m f^t_{\xi}(\mathbf{s}) \tag{17}$$

The goal is to minimize ϵ-approximation on latent distance $d(f_{\bar{\xi}}(\mathbf{s}), f_{\xi}(\mathbf{s}'))$ such that the left side of Eq.(15) is minimized. Particularly, given l_p-norm as distance $d(\cdot,\cdot)$ at the timestep t, and a ReLU network as function f, it leads to,

$$\begin{aligned}
\|f^t_{\bar{\xi}}(\mathbf{s}) - f^t_{\xi}(\mathbf{s}')\| &\leq \|\tau f^{t-1}_{\bar{\xi}}(\mathbf{s}) + (1-\tau) f^t_{\xi}(\mathbf{s}) - f^t_{\xi}(\mathbf{s}')\| \\
&= \|\tau f^{t-1}_{\bar{\xi}}(\mathbf{s}) - \tau f^t_{\xi}(\mathbf{s}) + f^t_{\xi}(\mathbf{s}) - f^t_{\xi}(\mathbf{s}')\| \\
&\leq \tau \|f^{t-1}_{\bar{\xi}}(\mathbf{s}) - f^t_{\xi}(\mathbf{s})\| + \|f^t_{\xi}(\mathbf{s}) - f^t_{\xi}(\mathbf{s}')\| \\
&\leq \tau \|f^{t-1}_{\bar{\xi}}(\mathbf{s}) - f^{t-1}_{\xi}(\mathbf{s}')\| + \tau \|f^{t-1}_{\xi}(\mathbf{s}') - f^t_{\xi}(\mathbf{s})\| + \|f^t_{\xi}(\mathbf{s}) - f^t_{\xi}(\mathbf{s}')\|
\end{aligned} \tag{18}$$

where $\tau = 1 - \tau_m$. L_f-Lipschitzness and L_g-Lipschitzness assumptions are employed as stated in Assumption A.1 and A.2. Suppose the updating has achieved convergence, Eq.(18) turns to the following inequality,

$$\|f_{\bar{\xi}}(\mathbf{s}) - f_{\xi}(\mathbf{s}')\| \leq \frac{1+\tau}{1-\tau} \|f_{\xi}(\mathbf{s}) - f_{\xi}(\mathbf{s}')\| \leq \frac{(1+\tau) L_f}{1-\tau} \|g_{\xi}(\mathbf{x}) - g_{\xi}(\mathbf{x}')\| \leq \frac{(1+\tau) L_f L_g}{1-\tau} \|\mathbf{x} - \mathbf{x}'\| \tag{19}$$

On the other hand, the second term on the right side of Eq.(15) can be rewritten as,

$$\|f_{\xi} \circ g_{\xi}(\mathbf{x}) - f_{\xi} \circ g_{\xi}(\mathbf{x}')\| \leq \|\xi_f\| \|g_{\xi}(\mathbf{x}) - g_{\xi}(\mathbf{x}')\| \leq L_g \|\xi_f\| \|\mathbf{x} - \mathbf{x}'\| \tag{20}$$

Altogether, we substitute Eq.(19) and Eq.(20) in Eq.(15), we finally obtain,

$$\mathbb{E}_{\mathbf{x}} \left[d(f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x}), f_{\xi} \circ g_{\xi}(\mathbf{x}')) \right] \leq \left(L_g \|\xi_f\| + \frac{(1+\tau) L_f L_g}{1-\tau} \right) \mathbb{E}_{\mathbf{x}} \left[\|\mathbf{x} - \mathbf{x}'\| \right] \tag{21}$$

The proof is finished. □

Theorem A.2. (Mixed CoIT.) Suppose that Lipschitzness holds for functions g_{ξ}, $g_{\bar{\xi}}$, f_{ξ} and $f_{\bar{\xi}}$, respectively. The updating dynamics is: $\bar{\xi}^t = (1-\tau_m)\bar{\xi}^{t-1} + \tau_m \xi^t$. For any input $\mathbf{x} \sim \hat{\mathcal{D}}$ and shifted $\mathbf{x}' \sim \mathcal{G}$, the divergence with mixed augmented states can be bound by,

$$\mathbb{E}_{\mathbf{x}} \left[d(f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x}), f_{\xi} \circ g_{\xi}(\mathbb{E}_{\mathbf{x}'}[\mathbf{x}'])) \right] \leq \rho \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{x}'} \left[||\mathbf{x} - \mathbf{x}'|| \right] \tag{22}$$

where $\rho = L_g \left(C L_f + \|\xi_f\| \right), C = \frac{1+\tau}{1-\tau}, \tau = 1 - \tau_m$. L_f and L_g are Lipschitz constants of the functions $f(\mathbf{s})$ and $g(\mathbf{x})$ respectively

Proof. Based on Theorem A.1, we can view the mixup $\mathbb{E}_{\mathbf{x}'}[\mathbf{x}']$ as a sort of transformed data, and thereby we have,

$$\mathbb{E}_{\mathbf{x}} \left[d(f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x}), f_{\xi} \circ g_{\xi}(\mathbb{E}_{\mathbf{x}'}[\mathbf{x}'])) \right] \leq L_g \left(C L_f + \|\xi_f\| \right) \mathbb{E}_{\mathbf{x}} \left[\|\mathbf{x} - \mathbb{E}_{\mathbf{x}'}[\mathbf{x}']\| \right] \tag{23}$$

Since $\mathbb{E}_{\mathbf{x}} \left[\|\mathbf{x} - \mathbb{E}_{\mathbf{x}'}[\mathbf{x}']\| \right] = \mathbb{E}_{\mathbf{x}} \left[\|\mathbb{E}_{\mathbf{x}'}[\mathbf{x} - \mathbf{x}']\| \right] \leq \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{x}'} \left[\|\mathbf{x} - \mathbf{x}'\| \right]$, we can finally obtain,

$$\mathbb{E}_{\mathbf{x}} \left[d(f_{\bar{\xi}} \circ g_{\bar{\xi}}(\mathbf{x}), f_{\xi} \circ g_{\xi}(\mathbb{E}_{\mathbf{x}'}[\mathbf{x}'])) \right] \leq L_g \left(C L_f + \|\xi_f\| \right) \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{x}'} \left[\|\mathbf{x} - \mathbf{x}'\| \right] \tag{24}$$

which completes the proof. □

B PSEUDOCODE

Algorithm 2 CoIT. Similarity metric. Learnable data transformation.

Inputs:
STATE encoder g_ξ, policy π_ϕ, Q-functions $Q_{\theta_1}, Q_{\theta_2}$, OBSERVATION encoder $g_{\bar{\xi}}$, PROJECTION f_ξ, $f_{\bar{\xi}}$
$\mu, \sigma \sim \mathcal{G}$ for TRANSFORM.
Scheduled standard deviation $\tilde{\sigma}(t)$ for the exploration noise
Training steps T, mini-batch size N, learning rate δ, target update rate τ, clip value c, TRANSFORM learning rate δ_{aug}, MOMENTUM update rate τ_m
Training:
for each timestep t in $1..T$ **do**
  $\tilde{\sigma}_t \leftarrow \tilde{\sigma}(t)$
  $\mathbf{x}'_t \leftarrow$ TRANSFORM$(\mathrm{x}_t, \mathcal{G}_t)$ and $\sigma_t \leftarrow 0$
  $\mathbf{a}_t \leftarrow \pi_\phi(g_\xi(\mathbf{x}'_t)) + \tilde{\epsilon}$ and $\tilde{\epsilon} \sim \mathcal{G}(0, \tilde{\sigma}_t)$
  $\mathbf{x}_{t+1} \sim P(\cdot|\mathbf{x}_t, \mathbf{a}_t)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{x}_t, \mathbf{a}_t, \mathcal{R}(\mathbf{x}_t, \mathbf{a}_t), \mathbf{x}_{t+1})$
  UPDATECRITIC$(\mathcal{D}, \tilde{\sigma}_t)$
  UPDATEACTOR$(\mathcal{D}, \tilde{\sigma}_t)$
end for
procedure UpdateCritic$(\mathcal{D}, \tilde{\sigma})$
  $\{(\mathbf{x}_t, \mathbf{a}_t, r_{t:t+n-1}, \mathbf{x}_{t+n})\}_{i=1}^N \sim \mathcal{D}$
  $\mathbf{x}'_t, \mathbf{x}'_{t+n} \leftarrow$ TRANSFORM$(\mathrm{x}_t, \mathcal{G}_t)$, TRANSFORM$(\mathrm{x}_{t+n}, \mathcal{G}_t)$
  $\mathbf{s}_t, \mathbf{s}_{t+n} \leftarrow g_\xi(\mathbf{x}'_t), g_\xi(\mathbf{x}'_{t+n})$
  $\mathbf{s}_{\bar{\xi}} \leftarrow g_{\bar{\xi}}(\mathbf{x}_t)$
  Measure similarity by $\mathcal{L}_{\xi,\bar{\xi},\omega}$
  $\mathbf{a}_{t+n} \leftarrow \pi_\phi(\mathbf{s}_{t+n}) + \tilde{\epsilon}$ and $\tilde{\epsilon} \sim \mathcal{G}(0, \tilde{\sigma}_t)$
  Compute $J_{\theta,\omega}(\mathcal{D})$ for Q_θ and TRANSFORM updating
  $\mu \leftarrow \mu - \delta_{\text{aug}} \nabla_\mu (J_{\theta,\omega,\xi}(\mathcal{D}))$ `# Learning` μ_t, `Line 6 in Algorithm 1.`
  $\sigma \leftarrow \sigma - \delta_{\text{aug}} \nabla_\sigma (J_{\theta,\omega,\xi}(\mathcal{D}))$ `# Learning` σ_t, `Line 6 in Algorithm 1.`
  $\xi \leftarrow \xi - \delta \nabla_\xi J_{\theta,\omega,\xi}(\mathcal{D})$
  $\theta \leftarrow \theta - \delta \nabla_\theta J_{\theta,\omega,\xi}(\mathcal{D})$
  $\hat{\theta} \leftarrow (1-\tau)\hat{\theta} + \tau\theta$
  $\bar{\xi} \leftarrow (1-\tau_m)\bar{\xi} + \tau_m \xi$
end procedure
procedure UpdateActor$(\mathcal{D}, \tilde{\sigma})$
  $\{\mathbf{x}_t\}_{i=1}^N \sim \mathcal{D}$
  $\mathbf{s}_t \leftarrow g_\xi(\text{TRANSFORM}(\mathbf{x}_t, \mathcal{G}_t))$
  $\mathbf{a}_t \leftarrow \pi_\phi(\mathbf{s}_t) + \tilde{\epsilon}$ and $\tilde{\epsilon} \sim \text{clip}(\mathcal{G}(0, \tilde{\sigma}))$
  Update the actor using the sampled policy gradient
  $\nabla_\phi J \approx \frac{1}{N} \sum_i \nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})|_{\mathbf{s}=\mathbf{s}_t, \mathbf{a}=\mathbf{a}_t} \nabla_\phi \pi(\mathbf{s})|_{\mathbf{s}_t}$
  $\phi \leftarrow \phi - \delta \nabla_\phi J$
end procedure

Table 2: We evaluate CoIT on the DMControl at 500k and 100k steps, compared with 4 baselines. CoIT outperfoms on 8 of 7 tasks both in 500k and 100k steps.

500K Steps Scores	CoIT	DrQ-v2	CURL	DDPG	SAC
Cartpole swingup sparse	731 ± 95	733 ± 47	$\mathbf{753 \pm 38}$	584 ± 328	714 ± 25
Pendulum swingup	$\mathbf{828 \pm 2}$	814 ± 12	53 ± 56	641 ± 352	488 ± 445
Hopper stand	$\mathbf{907 \pm 29}$	886 ± 44	663 ± 377	370 ± 340	390 ± 152
Walker walk	$\mathbf{958 \pm 10}$	758 ± 410	916 ± 17	118 ± 146	107 ± 74
Quadruped walk	$\mathbf{754 \pm 56}$	657 ± 133	272 ± 215	126 ± 70	32 ± 12
Finger turn hard	$\mathbf{509 \pm 126}$	271 ± 240	121 ± 88	73 ± 46	60 ± 42
Hopper hop	$\mathbf{386 \pm 81}$	176 ± 97	135 ± 88	79 ± 72	11 ± 24
Walker run	$\mathbf{585 \pm 47}$	524 ± 118	394 ± 35	28 ± 3	71 ± 28
100K Steps Scores					
Cartpole swingup sparse	$\mathbf{697 \pm 108}$	277 ± 380	480 ± 293	16 ± 36	21 ± 18
Pendulum swingup	$\mathbf{786 \pm 24}$	426 ± 392	3 ± 1	332 ± 358	221 ± 271
Hopper stand	$\mathbf{627 \pm 291}$	305 ± 339	460 ± 280	3 ± 3	252 ± 41
Walker walk	$\mathbf{491 \pm 258}$	321 ± 228	72 ± 93	465 ± 250	74 ± 74
Quadruped walk	$\mathbf{176 \pm 68}$	160 ± 66	147 ± 122	131 ± 28	54 ± 22
Finger turn hard	$\mathbf{223 \pm 87}$	58 ± 50	179 ± 135	73 ± 46	71 ± 36
Hopper hop	$\mathbf{179 \pm 35}$	34 ± 41	6 ± 10	2 ± 3	0 ± 0
Walker run	167 ± 11	201 ± 125	$\mathbf{257 \pm 15}$	26 ± 3	66 ± 24

C EXPERIMENTS

In this section, we explain the implementation details for CoIT in the DMControl setting. We use the DDPG as the backbone RL algorithm objective and build on top of the implementation from Yarats et al. (2021). We present in detail the hyperparameters for the architecture and optimization.

C.1 ACTOR AND CRITIC NETWORKS

The clipped double Q-learning (Van Hasselt et al., 2016; Fujimoto et al., 2018) is applied for the critic, where each Q-function is parametrized as a 3-layer MLP with `ReLU` activations after each layer except for the last. The actor is also a 3-layer MLP with `ReLU`s that outputs mean for the action. The hidden dimension is set to 1024 for both the critic and the actor.

C.2 ENCODER NETWORK

The architecture of the encoder is based on the work of Yarats et al. (2019), which has four convolutional layers with 3×3 kernels and 32 channels. The `ReLU` activation is applied after each conv layer. We also utilize `BatchNorm` (Ioffe & Szegedy, 2015) after each activation rather than `LayerNorm` (Ba et al., 2016) after a single fully-connected layer. The stride for the first conv layer is 2 while 1 for the rest. `BatchNorm` is also applied to normalize the fully-connected layer where the output of the convent is fed into. Finally, the use of `tanh` nonlinearity and the initialization of weight are consistent with the prior work. The actor and critic share the same encoder, although the encoder only uses the gradients from the critic for updating.

C.3 PROJECTION NETWORK

To encode the feature map into a latent space for similarity metrics, we introduce a projection network with its momentum version. The projection network is a 2-layer MLP with `ReLU` activations after the first layer. We set the hidden dimension as 1024 for each layer and the output dimension for contrastive loss is 128.

C.4 TRANSFORMATION DETAILS

Following prior works, we stack n 84×84 RGB images as observations that involve the temporal information to present the underlying state, where n is 4 in Atari100K and n is 3 in DMControl. For transformation, we first expand the images to 92×92 by padding 4 pixels at each side and then use `grid_sample` with the TRANSFORM operator to interpolate them to the original size.

C.5 HYPERPARAMETERS

For fair comparisons, our hyper-parameters are as consistent with Yarats et al. (2021) as possible. CoIT introduces two new hyperparameters α and λ to scale $\mathcal{K}_\omega(\mathbf{x}'_t)$ and $\mathcal{L}_{\xi,\bar{\xi},\omega}(\mathcal{D})$ in magnitude. The overview of used hyper-parameters in Atari100K and DeepMind Control Suite is shown in Table 4 and Table 5. We also present the learning rate of `TRANSFORM` and hyper-parameters of Eq. 11 for Atari100K in Table 3.

Table 3: An overview of used hyper-parameters for TRANSFORM of CoIT in Atari100K.

Game	TRANSFORM-lr	α	λ
Alien	$5e-07$	0.01	$5e-03$
Amidar	$2e-06$	0.01	$5e-03$
Assault	$2e-05$	$2e-03$	$5e-03$
Asterix	$2e-05$	$1e-03$	$5e-03$
Bank Heist	$2e-05$	0.01	$5e-03$
Battle Zone	$1e-05$	$1e-03$	$5e-03$
Boxing	$5e-07$	$1e-03$	$1e-03$
Breakout	$1e-05$	0.01	$2e-03$
Chopper Command	$1e-05$	0.01	$5e-03$
Crazy Climber	$2e-05$	$2e-03$	0.01
Demon Attack	$1e-06$	$1e-03$	$5e-03$
Freeway	$5e-06$	0.01	$5e-03$
Frostbite	$1e-05$	$2e-03$	$5e-03$
Gopher	$5e-07$	0.01	$5e-03$
Hero	$5e-05$	0.01	$5e-03$
Jamesbond	$2e-04$	0.01	$5e-03$
Kangaroo	$5e-06$	0.01	$5e-03$
Krull	$2e-06$	$5e-04$	$5e-03$
Kung Fu Master	$1e-04$	$1e-03$	0.05
Ms Pacman	$5e-07$	0.01	$5e-03$
Pong	$5e-07$	$5e-03$	0.02
Private Eye	$5e-07$	$1e-03$	$5e-03$
Qbert	$1e-05$	0.02	$5e-03$
Road Runner	$5e-06$	0.01	$5e-03$
Seaquest	$1e-06$	$5e-04$	0.01
Up N Down	$5e-06$	0.01	$5e-03$

C.6 MIXED CONTRASTIVE INVARIANT TRANSFORMATION

To demonstrate the effect of stabilizing the reward function by the Theorem 4.2, we present the results in Figure 5 and Figure 6 for comparisons between two versions of CoIT: (i) *w/o mix*. CoIT with single transformed data and (ii) *mix=2*. We use the transformation to produce 2 transformed data $\mathbf{x}_t^1, \mathbf{x}_t^2$, and mix them after encoding:

$$\mathbf{x}'_t = \eta \cdot \mathbf{x}_t^1 + (1-\eta) \cdot \mathbf{x}_t^2 \tag{25}$$

where η is a parameter randomly sampled from $(0, 1)$ and g_ξ denotes the online encoder.

In most scenarios, they gain similar performance with excellent stability. However, in the situation with sparse reward settings like Cartpole Swingup Sparse, CoIT without mixed data becomes extremely unstable, while CoIT for mixing 2 transformed data still gains outstanding performance.

Table 4: An overview of used hyper-parameters in Atari100K experiments.

Hyperparameter	**Setting**
Image size	$(84, 84)$
Replay buffer capacity	10^5
Training frames	4×10^5
Training steps	10^5
Frame skip	4
Stacked frames	4
Action repeat	4
Replay period every	1
Encoder: channels	32, 64
Encoder: filter size	$5 \times 5, 5 \times 5$
Encoder: stride	$5, 5$
Encoder: hidden dim	256
Momentum (EMA for CoIT)	0.001
Non-linearity	ReLU
Reward Clipping	$[-1, 1]$
Multi-step return	20
Minimum replay size for sampling	1600
Max frames per episode	$108K$
Update	Distributional Double Q
Target Network Update Period	every 2000 updates
Support-of-Q-distribution	51 bins
Mini-batch size	32
Discount γ	0.99
Optimizer	Adam
Optimizer: learning rate	10^{-4}
Optimizer: $\beta 1$	0.9
Optimizer: $\beta 2$	0.999
Optimizer ϵ	0.000015
Max gradient norm	10
Exploration	Noisy Nets
Noisy nets parameter	0.1
Priority exponent	0.5
Priority correction	$0.4 \rightarrow 1$
Critic Q-function soft-update rate τ	0.01
Similarity dim.	128

Table 5: An overview of used hyper-parameters in DeepMind Control Suite experiments.

Hyperparameter	**Setting**
Image size	$(84, 84)$
Replay buffer capacity	10^6
Stacked frames	3
Action repeat	Hopper Hop:4
	2
Seed frames	4000
Exploration steps	2000
n-step returns	3
Mini-batch size	256
Discount γ	0.99
Optimizer	Adam
Learning rate	10^{-4}
Augmentation learning rate	Hopper Hop: 1×10^{-6}
	2×10^{-6}
Agent update frequency	2
Critic Q-function soft-update rate τ	0.01
Momentum τ_m	0.0001
α	0.01
λ	0.005
Features dim.	50
Hidden dim.	1024
Similarity dim.	128
Exploration stddev. clip	0.3
Exploration stddev. schedule	linear$(1.0, 0.1, 100000)$ for 1M frames
	linear$(1.0, 0.1, 500000)$ for 3M frames

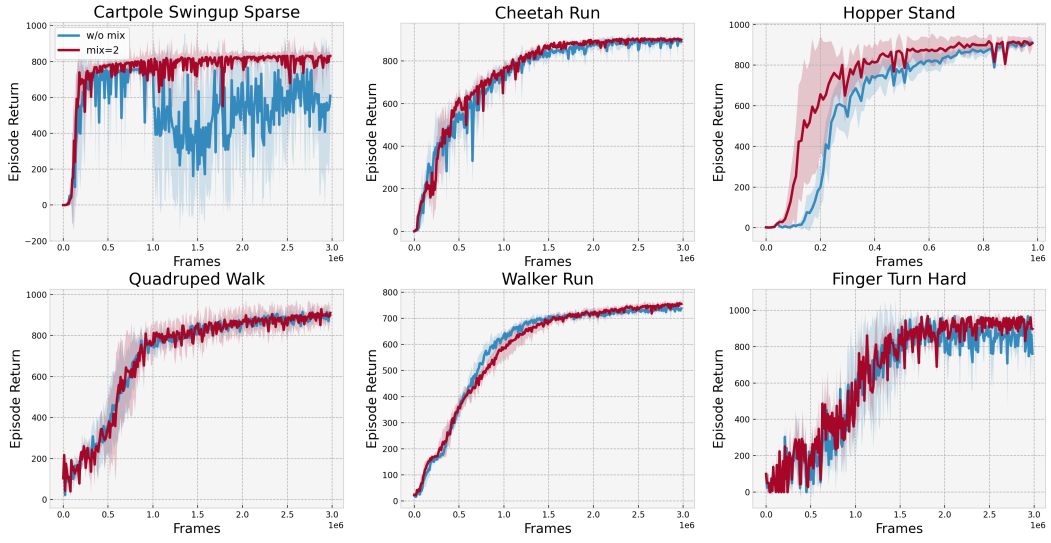

Figure 5: Comparisons between CoIT and its variants of without mix on the DMControl.

This appearance demonstrates that our mixed CoIT greatly stabilizes the training process, especially in sparse reward settings.

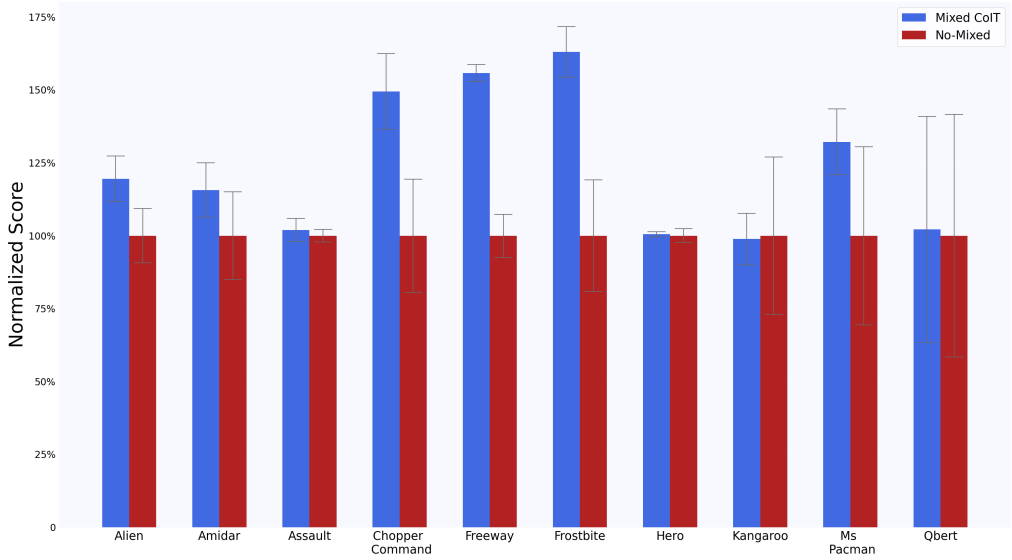

Figure 6: Comparisons between CoIT and its variants of without mix on Atari100K. To show the improvement of data mixing, we normalized the score of the no-mixed variants and plot the *mean* and *std*. The results are based on 10 runs of the outperformed tasks via CoIT.

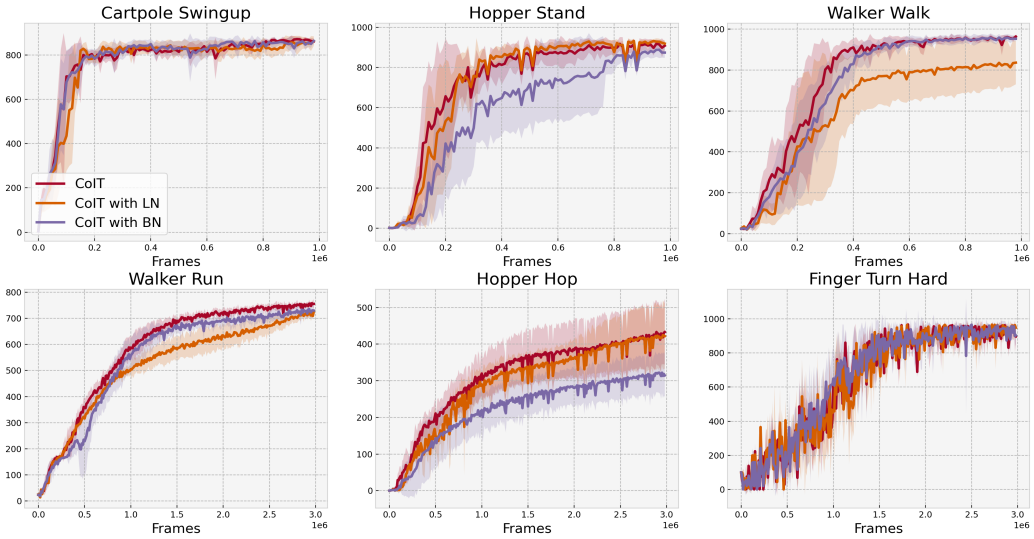

Figure 7: Comparisons between CoIT and its variants with different normalization for projection.

C.7 Types of Normalization for Projection

In this section, we compare CoIT to its two variants with different types of normalization in the projection network for similarity metrics, (i) *CoIT with LN* which using the `LayerNorm` after the first fully-connected layer, (ii) *CoIT with BN* which utilizing the `BatchNorm` after every fully-connected layer. We conduct experiments on 6 pixel-based continuous control tasks in the DMControl and present the results in Figure 7. Here we see that the original CoIT outperforms on each task and the performance of CoIT with BN is pretty close to the original CoIT. There is a score gap between the CoIT with LN and the original CoIT. This indicates that the specifically designed network architecture is beneficial to the RL performance.

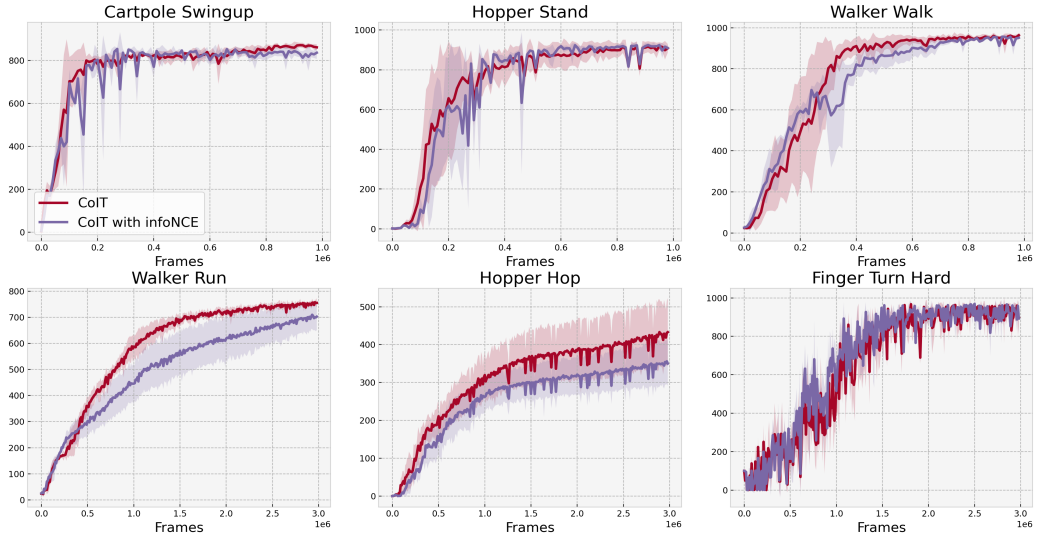

Figure 8: Comparisons between CoIT and its variants with different contrastive loss.

C.8 Types of Similarity Metric

We choose different types of contrastive loss for similarity metric and present results in Figure 8. We utilize the infoNCE loss which is widely used in self-supervised learning (He et al., 2020; Chen et al., 2020) and conduct a variant called *CoIT with infoNCE*. As shown in Figure 8, the original which optimizes the contrastive loss proposed in Grill et al. (2020) outperforms on each task in the DMControl. This indicates that naively making negative samples dissimilar may not be soundness.

C.9 Effects of Different Components in Object Function

Here we conduct experiments to study the effects of different components in object function for CoIT on extra 4 tasks in Figure 9. As mentioned before, we divide CoIT into 4 versions and show the performance impact of two auxiliary losses in CoIT separately: (i) *Critic*. Transformation is only updated with the critic. (ii) *X-stats & Critic*. Transformation is updated by critic and $\mathcal{K}_\omega(\mathbf{x}'_t)$ together. (iii) *H-dist & Critic*. Transformation is updated by critic and $\mathcal{L}_{\xi,\bar{\xi},\omega}$ together. (iv) *Unified Objective*. The transformation will be updated by all components.

From the curves, we demonstrate that both $\mathcal{K}_\omega(\mathbf{x}'_t)$ and $\mathcal{L}_{\xi,\bar{\xi},\omega}$ are sufficient for data-efficiency improvement and combining them together achieves remarkable performance and stability. We also find that *Critic* is facing performance degradation and is hard to improve the asymptotical performance in some domains. We consider that during the end of training the Gaussian distribution $\mathcal{G}_t(\mu_t, \sigma_t)$ converges to a small boundary, thus the produced virtual data is augmented in a slight amplitude. This makes the CoIT run into a situation similar to vanilla SAC & DDPG: updating by reward signal purely is quite unfavorable for representation learning as the agent focuses on slight reward-relevant information. For this reason, we introduce $\mathcal{K}_\omega(\mathbf{x}'_t)$ and $\mathcal{L}_{\xi,\bar{\xi},\omega}$ to assist the agent to learn meaningful representations and stabilize the reward function.

C.10 Saliency Maps

To better present the representations learned by CoIT, we present the saliency maps on the augmented data (e.g., *Random Shift*) of the encoder for 6 tasks on the DMControl in Figure 10.

These saliency maps demonstrate that CoIT is beneficial for the agent to focus on task-relevant elements like the whole robot body and ignore the task-irrelevant information like the floor and background. What's more, the agent trained with CoIT pays high attention to the reward-relevant signals while still taking note of other components that are useful for the task at hand. In Finger Turn Hard, for instance, the lightest part in the saliency map is a red ball in the observation, which

Figure 9: Experiments for studying the effects of different components in object function.

is highly related to the reward. However, the agent also focuses on the robot's body for reasonable action taking. This appearance inspires us that excellent representation learning is a trade-off: the agent needs to figure out which elements are highly related to the reward while still concerned with as much information as possible.

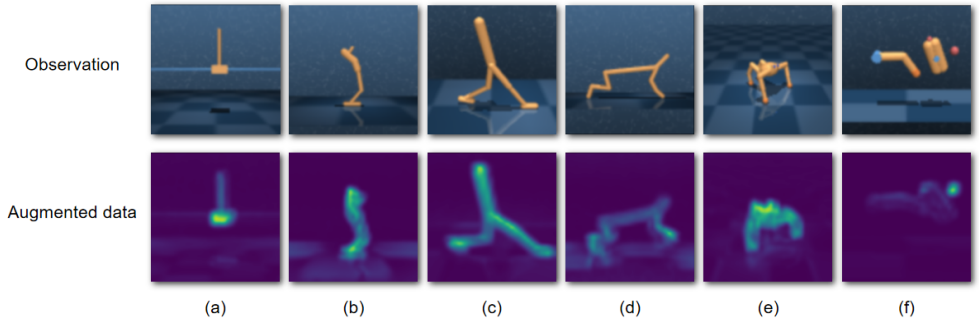

Figure 10: Saliency maps for CoIT on 6 tasks: (a) Cartpole Swingup Sparse, (b) Hopper Stand, (c) Walker Run, (d) Cheetah Run, (e) Quadruped Walk, and (f) Finger Turn Hard.

D Connection to Batch Normalization

Batch normalization (BatchNorm) aims to alleviate the shift caused by the randomness of input data for neural network layers, which is called as internal covariate shift. In particular, the change of *means* and *variances* of the distributions in each layer during mini-batch training creates the problem that parameters are sensitive to the initialization and required to consistently adapt to the distributions. Normalization by batch statistics employs the empirical mean and variance on the feature scalar,

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i, \quad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{x}_i - \mu_B\right)^2 \tag{26}$$

where B denotes a mini-batch of size m in training. Then, the feature dimension of each layer is normalized by,

$$\hat{\mathbf{x}}_i^{(k)} = \frac{\mathbf{x}_i^{(k)} - \mu_B^{(k)}}{\sqrt{{\sigma_B^{(k)}}^2 + \epsilon}} \tag{27}$$

where ϵ is a small constant added for numerical stability, $k \in [1, d]$ is the index of feature dimensions d, and i is the index of data points. A counterpart procedure of the inner batch normalization is

whitening which is non-differential everywhere making the mean of 0 and the variance of 1 for the distribution. In turn, given the distribution with mean μ_B and variance σ_B^2, it is straightforward that the deviation given by $\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|$ is associated with the value of $\mathbf{x}_i$ and the collection of $w_B = \{\mu_B, \sigma_B\}$.

Different with the goal of whitening procedure with 0-mean and 1-variance, the automatic data augmentation needs to find an optimal interpolation $\mathbf{x}'$ obey certain distribution with unknown mean μ and variance σ^2. As the given environment is non-stationary, the distribution for data generation is shifted with respect to the parameter sets $w_t = \{\mu_t, \sigma_t\}$, where t denotes a timestep. To cope with the assumption of a stationary environment using the replay buffer in off-policy methods, the automatic data augmentation allows a new dynamic distribution controlling by the model-free RL. In other words, the dynamics of the normalization derive from the observation distribution conditioned on the changed environment.

Concretely, the augmented data $\mathbf{x}' \sim \hat{\mathcal{D}}'$ is parameterized through the Gaussian distribution $\mathcal{G}(\mu, \sigma)$, as the Algorithm 1 shown. Similar to BatchNorm, $\|\mathbf{x} - \mathbf{x}'\|$ is determined by the value of the original image $\mathbf{x}$ and the dynamically changed $\{\mu_t, \sigma_t\}$. As a consequence, a normalized shift distribution conditioned on the values of the replay buffer is obtained for the current training round.