CURL:Contrastive Unsupervised Representations for Reinforcement Learning

Aravind Srinivas Michael Laskin Pieter Abbeel

Ece Ada

AUTHORS

Aravind Srinivas
Pieter Abbeel

Michael Laskin

Contributions

Contrastive leaning and model free RL with minimal changes in architecture and hyperparameter optimization

Purely from pixels

Self-supervised

Sample efficient

Contrastive leaning:” building differentiable dictionary lookups over high dimensional entities”

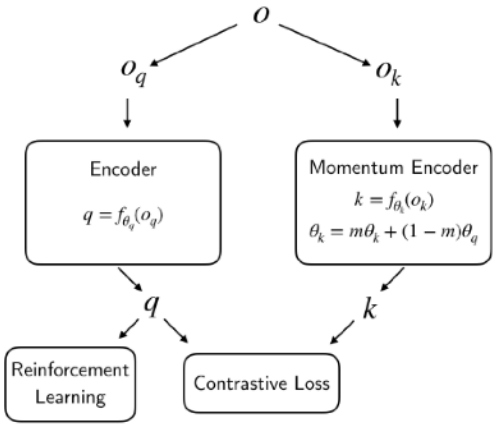

Figure 1. **C**ontrastive **U**nsupervised Representations for **R**einforcement **L**earning (CURL) combines instance contrastive learning and reinforcement learning. CURL trains a visual representation encoder by ensuring that the embeddings of data-augmented versions o_q and o_k of observation o match using a contrastive loss. The *query* observations o_q are treated as the anchor while the *key* observations o_k contain the positive and negatives, all constructed from the minibatch sampled for the RL update. The keys are encoded with a momentum averaged version of the query encoder. The RL policy and (or) value function are built on top of the query encoder which is jointly trained with the contrastive and reinforcement learning objectives. CURL is a generic framework that can be plugged into any RL algorithm that relies on learning representations from high dimensional images.

CURL Architecture

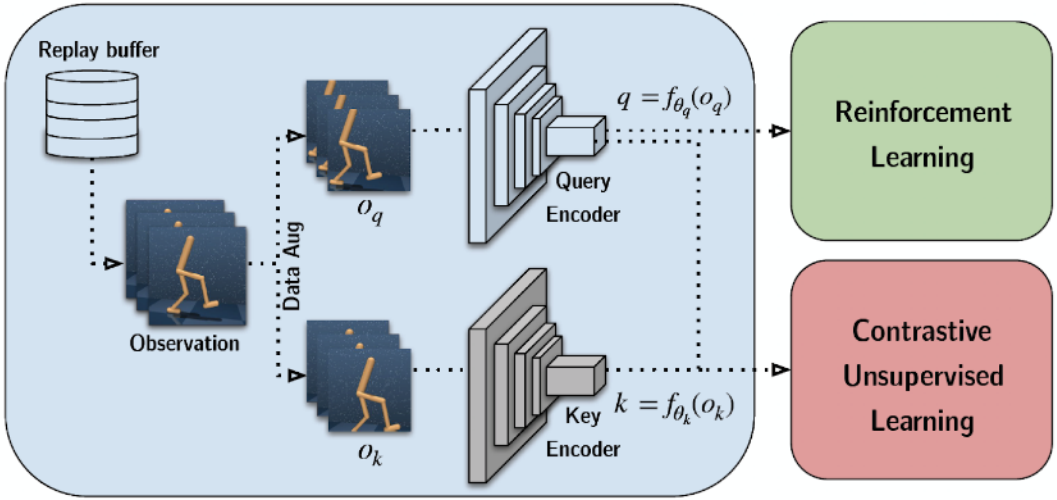

Figure 2. CURL Architecture: A batch of transitions is sampled from the replay buffer. Observations are then data-augmented twice to form *query* and *key* observations, which are then encoded with the query encoder and key encoders, respectively. The *queries* are passed to the RL algorithm while *query-key* pairs are passed to the contrastive learning objective. During the gradient update step, only the *query* encoder is updated. The *key* encoder weights are the moving average (EMA) of the query weights similar to MoCo (He et al., 2019a).

InfoNCE Loss

$$\mathcal{L}_q = \log \frac{\exp(q^T W k_+)}{\exp(q^T W k_+) + \sum_{i=0}^{K-1} \exp(q^T W k_i)} \qquad (4)$$

Similarities between anchor and targets are learned with bilinear products

Logloss of K-way softmax classifier with label k_+

Discrimination Objective

Simpler is better for RL due to instability

Contrastive instance discrimintation: maximizing the mutual information between image_1 and augmented_image1

InfoNCE loss requires minimal architectural adjustments

Query Key Pair Generation

Anchor and positive observations from same image

Negative observations from other images

Random crop data augmentation

Instance discrimination on frame stacks to learn spatio-temporal discriminative features.

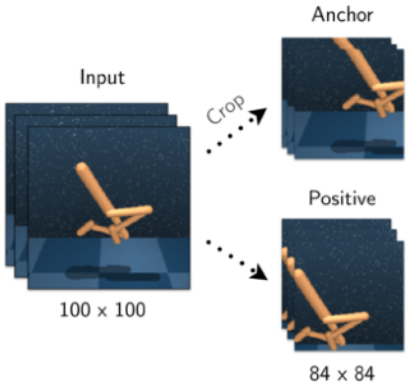

Figure 3. Visually illustrating the process of generating an anchor and its positive using stochastic random crops. Our aspect ratio for cropping is 0.84, i.e, we crop a 84×84 image from a 100×100 simulation-rendered image. Applying the same random crop coordinates across all frames in the stack ensures time-consistent spatial jittering.

Similarity Measure

$$\mathrm{sim}(q, k) = q^T W k$$

Bilinear inner product outperforms normalized dot product.

W=learned parameter matrix

EMA=Momentum averaged encoder

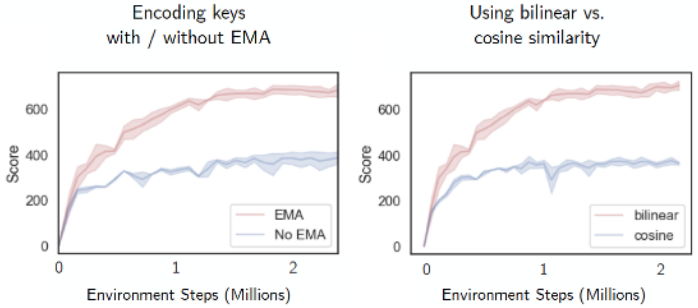

Figure 5. Performance on cheetah-run environment ablated two-ways: (left) using the query encoder or exponentially moving average of the query encoder for encoding keys (right) using the bi-linear inner product as in (van den Oord et al., 2018) or the cosine inner product as in He et al. (2019b); Chen et al. (2020)

Target Encoding with Momentum

High dimensional pixels to
expressive semantic latent features

Stop gradient f_k

Exponentially moving average

$$\theta_k = m\theta_k + (1 - m)\theta_q$$

Important:

Increasing size of dictionary

Enriching set of negatives

Pseudocode

```
def encode(x,z_dim):
    """
    ConvNet encoder
    args:
        B-batch_size, C-channels
        H,W-spatial_dims
        x : shape : [B, C, H, W]
        C = 3 * num_frames; 3 - R/G/B
        z_dim: latent dimension
    """

    x = x / 255.

    # c: channels, f: filters
    # k: kernel, s: stride

    z = Conv2d(c=x.shape[1], f=32, k=3, s=2)])(
        x)
    z = ReLU(z)

    for _ in range(num_layers - 1):
        z = Conv2d((c=32, f=32, k=3, s=1))(z)
        z = ReLU(z)

    z = flatten(z)

    # in: input dim, out: output_dim, h:
        hiddens

    z = mlp(in=z.size(),out=z_dim,h=1024)
    z = LayerNorm(z)
    z = tanh(z)
```

```
# f_q, f_k: encoder networks for anchor
# (query) and target (keys) respectively.
# loader: minibatch sampler from ReplayBuffer
# B-batch_size, C-channels, H,W-spatial_dims
# x : shape : [B, C, H, W]
# C = c * num_frames; c=3 (R/G/B) or 1 (gray)
# m: momentum, e.g. 0.95
# z_dim: latent dimension
f_k.params = f_q.params
W = rand(z_dim, z_dim) # bilinear product.
for x in loader: # load minibatch from buffer
 x_q = aug(x) # random augmentation
 x_k = aug(x) # different random augmentation
 z_q = f_q.forward(x_q)
 z_k = f_k.forward(x_k)
 z_k = z_k.detach() # stop gradient
 proj_k = matmul(W, z_k.T) # bilinear product
 logits = matmul(z_q, proj_k) # B x B
  # subtract max from logits for stability
 logits = logits - max(logits, axis=1)
 labels = arange(logits.shape[0])
 loss = CrossEntropyLoss(logits, labels)
 loss.backward()
 update(f_q.params) # Adam
 update(W) # Adam
 f_k.params = m*f_k.params+(1-m)*f_q.params
```

Experiments

https://www.youtube.com/watch?time_continue=11&v=a6SyIg4HrbQ&feature=emb_logo

Table 1. Scores achieved by CURL and baselines on DMControl500k and 1DMControl100k. CURL achieves state-of-the-art performance on the majority (**5** out of **6**) environments benchmarked on DMControl500k. These environments were selected based on availability of data from baseline methods (we run CURL experiments on 16 environments in total and show results in Figure 7). The baselines are PlaNet (Hafner et al., 2018), Dreamer (Hafner et al., 2019), SAC+AE (Yarats et al., 2019), SLAC (Lee et al., 2019), pixel-based SAC and state-based SAC (Haarnoja et al., 2018). SLAC results were reported with one and three gradient updates per agent step, which we refer to as SLACv1 and SLACv2 respectively. We compare to SLACv1 since all other baselines and CURL only make one gradient update per agent step. We also ran CURL with three gradient updates per step and compare results to SLACv2 in Table 5.

500K STEP SCORES	CURL	PLANET	DREAMER	SAC+AE	SLACV1	PIXEL SAC	STATE SAC
FINGER, SPIN	**926 ± 45**	561 ± 284	796 ± 183	884 ± 128	673 ± 92	179 ± 166	923 ± 21
CARTPOLE, SWINGUP	**841 ± 45**	475 ± 71	762 ± 27	735 ± 63	-	419 ± 40	848 ± 15
REACHER, EASY	**929 ± 44**	210 ± 390	793 ± 164	627 ± 58	-	145 ± 30	923 ± 24
CHEETAH, RUN	518 ± 28	305 ± 131	**732 ± 103**	550 ± 34	640 ± 19	197 ± 15	795 ± 30
WALKER, WALK	**902 ± 43**	351 ± 58	897 ± 49	847 ± 48	842 ± 51	42 ± 12	948 ± 54
BALL IN CUP, CATCH	**959 ± 27**	460 ± 380	879 ± 87	794± 58	852 ± 71	312± 63	974 ± 33
100K STEP SCORES							
FINGER, SPIN	**767 ± 56**	136 ± 216	341 ± 70	740 ± 64	693 ± 141	179 ± 66	811±46
CARTPOLE, SWINGUP	**582±146**	297±39	326±27	311±11	-	419±40	835±22
REACHER, EASY	**538±233**	20±50	314±155	274±14	-	145±30	746±25
CHEETAH, RUN	299 ±48	138±88	238± 76	267±24	**319±56**	197±15	616±18
WALKER, WALK	**403±24**	224±48	277±12	394±22	361±73	42±12	891±82
BALL IN CUP, CATCH	**769 ± 43**	0 ± 0	246 ± 174	391± 82	512 ± 110	312± 63	746±91

Table 2. Scores achieved by CURL and baselines on Atari benchmarked at 100k time-steps (Atari100k). CURL achieves state-of-the-art performance on **14** out of **26** environments. Our baselines are SimPLe (Kaiser et al., 2019), OverTrained Rainbow (OTRainbow) (Kielak, 2020), Data-Efficient Rainbow (Eff. Rainbow) (van Hasselt et al., 2019), Rainbow (Hessel et al., 2017), Random Agent and Human Performance (Human). Prior work has reported different numbers for some of these baselines, particularly, SimPLe and Human. To be rigorous, we pick the best number for each game across the tables reported in Kielak (2020) and van Hasselt et al. (2019).

GAME	HUMAN	RANDOM	RAINBOW	SIMPLE	OTRAINBOW	EFF. RAINBOW	CURL
ALIEN	7127.7	227.8	318.7	616.9	824.7	739.9	**1148.2**
AMIDAR	1719.5	5.8	32.5	88.0	82.8	188.6	**232.3**
ASSAULT	1496	222.4	231	527.2	351.9	431.2	**543.7**
ASTERIX	8503.3	210.0	243.6	**1128.3**	628.5	470.8	524.3
BANK HEIST	753.1	14.2	15.55	34.2	182.1	51.0	**193.7**
BATTLE ZONE	37800	3285.71	2360.0	5184.4	4060.6	10124.6	**11208.0**
BOXING	12.1	0.1	-24.8	**9.1**	2.5	0.2	4.8
BREAKOUT	31.8	1.7	1.2	16.4	9.84	1.9	**18.2**
CHOPPER COMMAND	9882	811.0	120	**1246.9**	1033.33	861.8	1198.0
CRAZY_CLIMBER	35411	10780.5	2254.5	**62583.6**	21327.8	16185.3	27805.6
DEMON_ATTACK	3401	152.1	163.6	208.1	711.8	508.0	**834.0**
FREEWAY	29.6	0.0	0.0	20.3	25	**27.9**	**27.9**
FROSTBITE	4334.7	65.2	60.2	254.7	231.6	866.8	**924.0**
GOPHER	2412.5	257.6	431.2	771.0	778	349.5	**801.4**
HERO	30826.4	1027.0	487	2656.6	6458.8	**6857.0**	6235.1
JAMESBOND	406.7	29.0	47.4	125.3	112.3	301.6	**400.1**
KANGAROO	3035.0	52.0	0.0	323.1	605.4	779.3	**345.3**
KRULL	2665.5	1598.0	1468	**4539.9**	3277.9	2851.5	3833.6
KUNG_FU_MASTER	22736.3	258.5	0.	**17257.2**	5722.2	14346.1	14280
MS_PACMAN	15693	307.3	67	1480	941.9	1204.1	**1492.8**
PONG	14.6	-20.7	-20.6	**12.8**	1.3	-19.3	2.1
PRIVATE EYE	69571.3	24.9	0	58.3	100	97.8	**105.2**
QBERT	13455.0	163.9	123.46	**1288.8**	509.3	1152.9	1225.6
ROAD_RUNNER	7845.0	11.5	1588.46	5640.6	2696.7	**9600.0**	6786.7
SEAQUEST	42054.7	68.4	131.69	**683.3**	286.92	354.1	408
UP_N_DOWN	11693.2	533.4	504.6	**3350.3**	2847.6	2877.4	2735.2

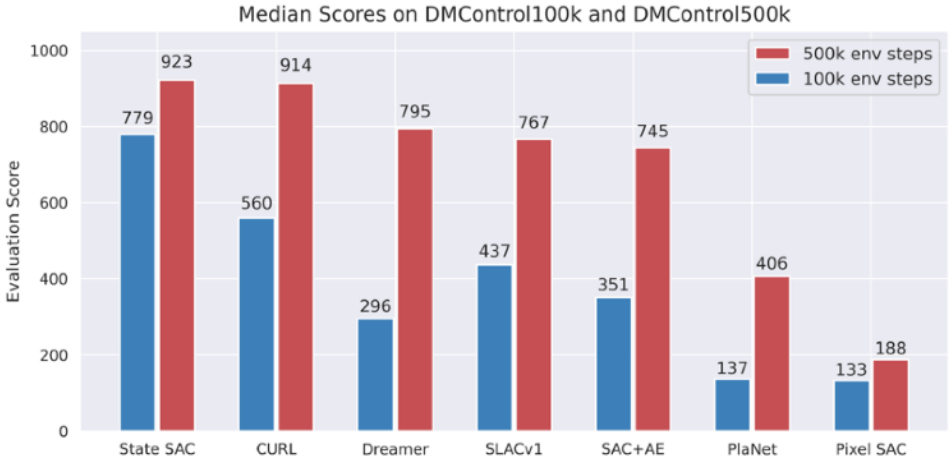

Figure 4. Performance of CURL coupled to SAC averaged across 10 seeds relative to SLACv1, PlaNet, Pixel SAC and State SAC baselines. At the 500k benchmark CURL matches the median score of state-based SAC. At 100k environment steps CURL achieves a 1.9x higher median score than Dreamer. For a direct comparison, we only compute the median across the 6 environments in 1 (4 for SLAC) and show learning curves for CURL across 16 DMControl experiments in 7.

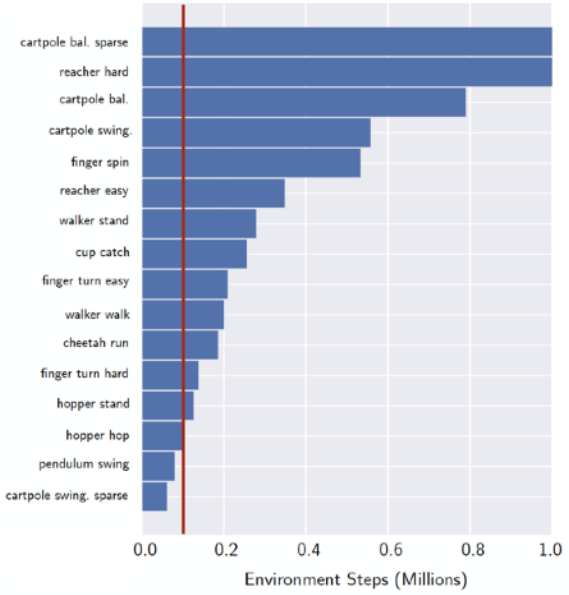

Figure 6. The number of steps it takes a prior leading pixel-based method, Dreamer, to achieve the same score that CURL achieves at 100k training steps (clipped at 1M steps). On average, CURL is 4.5x more data-efficient. We chose Dreamer because the authors (Hafner et al., 2019) report performance for all of the above environments while other baselines like SLAC and SAC+AE only benchmark on 4 and 6 environments, respectively. For further comparison of CURL with these methods, the reader is referred to Table 1 and Figure 4.

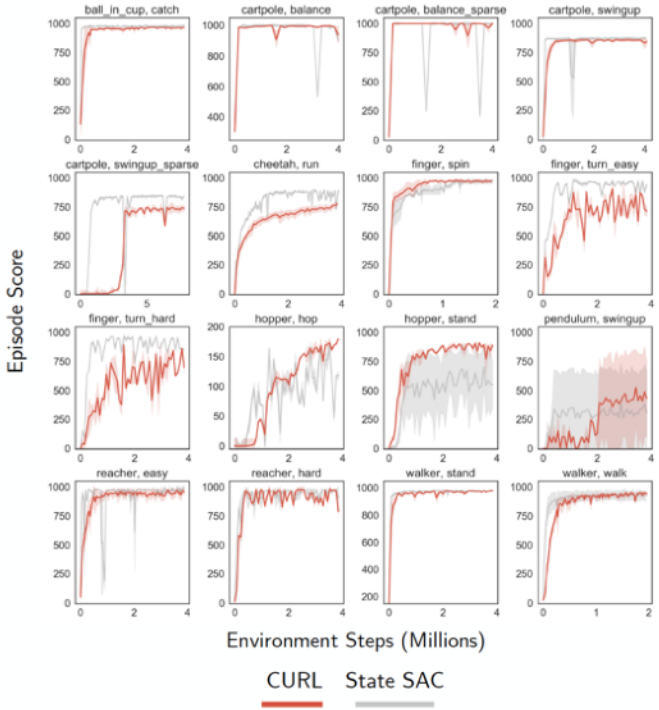

Figure 7. CURL compared to state-based SAC run for 2 seeds on each of 16 selected DMControl environments. For the 6 environments in 4, CURL performance is averaged over 10 seeds.

Predicting States from Pixels

Figure 10. Test-time mean squared error for predicting the proprioceptive state from pixels on a number of DMControl environments. In DMControl, environments fall into two groups - where the state corresponds to either (a) positions and velocities of the robot joints or (b) the joint angles and angular velocities.

Thank you! ☺️