Data Augmentation for RL

and why now?
Data augmentation in images

(a) Original  (b) Crop and resize  (c) Crop, resize (and flip)  (d) Color distort. (drop)  (e) Color distort. (jitter)

(f) Rotate \(\{90^\circ, 180^\circ, 270^\circ\}\)  (g) Cutout  (h) Gaussian noise  (i) Gaussian blur  (j) Sobel filtering
Data augmentation in RL

- Domain randomization
- Cutout
- Random convolutions
Data augmentations in contrastive learning

MoCo

SwAV

- MoCo (Momentum Contrastive Learning)
  - encoder
  - query: $x^{\text{query}}$
  - queue: $k_0, k_1, k_2, \ldots$
  - contrastive loss
  - similarity: $q$

- SwAV (Swapping Augmented Views)
  - $X$ (input)
  - $f_\theta$ (encoder)
  - $z_1, z_2$ (encodings)
  - Codes
  - Swapped Prediction
  - Prototypes: $c$
  - (b) Crop and resize
  - (c) Crop, resize (and flip)
  - (d) Color distort. (drop)
  - (g) Cutout
  - (h) Gaussian noise
  - (i) Gaussian blur
Part 1 - CURL*

\[ q = f_{\theta_q}(o_q) \]

\[ k = f_{\theta_k}(o_k) \]

\[ \theta_k = m\theta_k + (1-m)\theta_q \]

Contrastive Loss

* CURL = Contrastive Unsupervised Representations for Reinforcement Learning
CURL

Positives = different random crops of the same observation

Negatives = Different observation

Loss = InfoNCE

\[
\mathcal{L}_q = \log \frac{\exp(q^T W k_+)}{\exp(q^T W k_+) + \sum_{i=0}^{K-1} \exp(q^T W k_i)}
\]
## Results

### 500K Step Scores

<table>
<thead>
<tr>
<th>Game</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINGER_SPIN</td>
<td>926 ± 45</td>
<td>561 ± 284</td>
<td>796 ± 183</td>
<td>884 ± 128</td>
<td>673 ± 92</td>
<td>179 ± 166</td>
<td>923 ± 31</td>
<td>923 ± 31</td>
<td>923 ± 31</td>
<td>923 ± 31</td>
<td>923 ± 31</td>
</tr>
<tr>
<td>CARTPOLE_SWINGUP</td>
<td>841 ± 45</td>
<td>475 ± 71</td>
<td>762 ± 27</td>
<td>735 ± 63</td>
<td>-</td>
<td>419 ± 40</td>
<td>848 ± 15</td>
<td>848 ± 15</td>
<td>848 ± 15</td>
<td>848 ± 15</td>
<td>848 ± 15</td>
</tr>
<tr>
<td>REACHER_EASY</td>
<td>929 ± 44</td>
<td>210 ± 390</td>
<td>793 ± 164</td>
<td>627 ± 58</td>
<td>-</td>
<td>145 ± 30</td>
<td>923 ± 24</td>
<td>923 ± 24</td>
<td>923 ± 24</td>
<td>923 ± 24</td>
<td>923 ± 24</td>
</tr>
<tr>
<td>CHEETAH_RUN</td>
<td>518 ± 28</td>
<td>305 ± 131</td>
<td>570 ± 253</td>
<td>550 ± 34</td>
<td>640 ± 19</td>
<td>197 ± 15</td>
<td>795 ± 30</td>
<td>795 ± 30</td>
<td>795 ± 30</td>
<td>795 ± 30</td>
<td>795 ± 30</td>
</tr>
<tr>
<td>WALKER_WALK</td>
<td>902 ± 43</td>
<td>351 ± 58</td>
<td>897 ± 49</td>
<td>847 ± 48</td>
<td>842 ± 51</td>
<td>42 ± 12</td>
<td>948 ± 54</td>
<td>948 ± 54</td>
<td>948 ± 54</td>
<td>948 ± 54</td>
<td>948 ± 54</td>
</tr>
<tr>
<td>BALL_IN_CUP_CATCH</td>
<td>959 ± 27</td>
<td>460 ± 380</td>
<td>879 ± 87</td>
<td>794 ± 58</td>
<td>852 ± 71</td>
<td>312 ± 63</td>
<td>974 ± 33</td>
<td>974 ± 33</td>
<td>974 ± 33</td>
<td>974 ± 33</td>
<td>974 ± 33</td>
</tr>
</tbody>
</table>

### 100K Step Scores

<table>
<thead>
<tr>
<th>Game</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINGER_SPIN</td>
<td>767 ± 56</td>
<td>136 ± 216</td>
<td>341 ± 70</td>
<td>740 ± 64</td>
<td>693 ± 141</td>
<td>179 ± 66</td>
<td>811 ± 46</td>
<td>811 ± 46</td>
<td>811 ± 46</td>
<td>811 ± 46</td>
<td>811 ± 46</td>
</tr>
<tr>
<td>CARTPOLE_SWINGUP</td>
<td>582 ± 146</td>
<td>297 ± 39</td>
<td>326 ± 27</td>
<td>311 ± 11</td>
<td>-</td>
<td>419 ± 40</td>
<td>835 ± 22</td>
<td>835 ± 22</td>
<td>835 ± 22</td>
<td>835 ± 22</td>
<td>835 ± 22</td>
</tr>
<tr>
<td>CHEETAH_RUN</td>
<td>299 ± 48</td>
<td>138 ± 88</td>
<td>235 ± 137</td>
<td>267 ± 24</td>
<td>319 ± 56</td>
<td>197 ± 15</td>
<td>616 ± 18</td>
<td>616 ± 18</td>
<td>616 ± 18</td>
<td>616 ± 18</td>
<td>616 ± 18</td>
</tr>
<tr>
<td>WALKER_WALK</td>
<td>403 ± 24</td>
<td>224 ± 48</td>
<td>277 ± 12</td>
<td>394 ± 22</td>
<td>361 ± 73</td>
<td>42 ± 12</td>
<td>891 ± 82</td>
<td>891 ± 82</td>
<td>891 ± 82</td>
<td>891 ± 82</td>
<td>891 ± 82</td>
</tr>
<tr>
<td>BALL_IN_CUP_CATCH</td>
<td>769 ± 43</td>
<td>0 ± 0</td>
<td>246 ± 174</td>
<td>391 ± 82</td>
<td>512 ± 110</td>
<td>312 ± 63</td>
<td>746 ± 91</td>
<td>746 ± 91</td>
<td>746 ± 91</td>
<td>746 ± 91</td>
<td>746 ± 91</td>
</tr>
</tbody>
</table>

### tl;dr

It works better than everything else.
Part 2 - DrQ/RAD*

*RAD = Reinforcement Learning with Augmented Data
DrQ = Data Regularized Q
Part 2 - DrQ/RAD*

*RAD = Reinforcement Learning with Augmented Data
DrQ = Data Regularized Q
RAD

Input

Crop

Translate

Window

Grayscale

Cutout

Flip

Rotate

Cutout-color

Random conv

Color-jitter

Augmentations applied consistently across stacked frames
Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Crop</th>
<th>Grayscale</th>
<th>Rotate</th>
<th>Cutout</th>
<th>Color-jitter</th>
<th>Flip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crop</td>
<td>920</td>
<td>849</td>
<td>635</td>
<td>855</td>
<td>797</td>
<td>650</td>
</tr>
<tr>
<td>Grayscale</td>
<td>856</td>
<td>175</td>
<td>349</td>
<td>187</td>
<td>214</td>
<td>231</td>
</tr>
<tr>
<td>Rotate</td>
<td>604</td>
<td>403</td>
<td>268</td>
<td>293</td>
<td>392</td>
<td>394</td>
</tr>
<tr>
<td>Cutout</td>
<td>722</td>
<td>206</td>
<td>376</td>
<td>215</td>
<td>31</td>
<td>284</td>
</tr>
<tr>
<td>Color-jitter</td>
<td>831</td>
<td>227</td>
<td>420</td>
<td>407</td>
<td>194</td>
<td>265</td>
</tr>
<tr>
<td>Flip</td>
<td>828</td>
<td>210</td>
<td>391</td>
<td>244</td>
<td>264</td>
<td>223</td>
</tr>
</tbody>
</table>

tl;dr It works better than everything else
Our approach, DrQ, is the union of the three separate regularization mechanisms introduced above:

1. transformations of the input image (Section 3.1).
2. averaging the $Q$ target over $K$ image transformations (Equation (1)).
3. averaging the $Q$ function itself over $M$ image transformations (Equation (3)).

$$y_i = r_i + \gamma \frac{1}{K} \sum_{k=1}^{K} Q_\theta(f(s'_i, \nu'_i, a'_{i,k})) \text{ where } a'_{i,k} \sim \pi(\cdot | f(s'_i, \nu'_i, a'_{i,k}))$$

$$\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \left( \frac{1}{NM} \sum_{i=1, m=1}^{N, M} (Q_\theta(f(s_i, \nu_i, a_i), a_i) - y_i)^2 \right).$$
Part 3 - DrAC*

\[ J_{PG}(\theta) = \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s) \hat{A}_{\theta_{old}}(s, a) = \mathbb{E}_{a \sim \pi_{\theta_{old}}} \left[ \frac{\pi_{\theta}(a|f(s))}{\pi_{\theta_{old}}(a|s)} \hat{A}_{\theta_{old}}(s, a) \right], \]

To ensure that the policy and value functions are invariant to these transformations of the input state, we propose an additional loss term for regularizing the policy,

\[ G_{\pi} = KL [\pi_{\theta}(a|f(s, \nu)) | \pi(a|s)] , \tag{3} \]

as well as an extra loss term for regularizing the value function,

\[ G_{V} = (V_{\theta}(f(s, \nu)) - V(s))^2 . \tag{4} \]

Thus, our data-regularized actor-critic method, or DrAC, maximizes the following objective:

\[ J_{DrAC} = J_{PPO} - \alpha_{r}(G_{\pi} + G_{V}) \tag{5} \]

*DrAC = Data-regularized Actor-Critic
DrAC

**Chaser**

- PPO
- RAD (random-conv)
- RAD (grayscale)
- DrAC (Ours) (random-conv)
- DrAC (Ours) (grayscale)

**Miner**

**StarPilot**
Discussion (not real)

Bad science

8 Apr 2020: CURL on arXiv
28 Apr 2020: DrQ on arXiv
30 Apr 2020: RAD on arXiv

u/rlbeaverton: “I digged through this code and I was unable to find any major difference between CURL and RAD except this commented out lines
https://github.com/MishaLaskin/rad/blob/master/curl_sac.py#L494-L496.”

WWYD?
Discussion

Is CURL useful at all?

Berkeley folk: “For example, RAD would not be applicable to environments with sparse-rewards or no rewards at all, because it learns similarity consistency implicitly through the observations coupled to a reward signal. Unsupervised representation learning may therefore be a better fit for real-world tasks, such as robotic manipulation, where the environment reward is more likely to be sparse or absent.”

True or False?
Discussion

Where does SSL/Contrastive learning in RL belong?

Why on-policy at all?
Discussion

Would contrastive loss (aux loss) ever help improve policy performance, or just speed up convergence?

Decoupling Representation Learning from Reinforcement Learning

Discussion

Why now?

**Berkeley folk:** “Over the last few years, two trends have converged to make data-efficient visual RL possible. First, end-to-end RL algorithms have become increasingly more stable through algorithms like the Rainbow DQN, TD3, and SAC. Second, there has been tremendous progress in label-efficient learning for image classification using contrastive unsupervised representations (CPCv2, MoCo, SimCLR) and data augmentation (MixUp, AutoAugment, RandAugment).”

A: I do not know.
Discussion

- Augmentations in state-space

  techniques: (a) **random amplitude scaling**, introduced in this work, multiplies the uniform random variable, i.e., \( s' = s \times z \), where \( z \sim \text{Uni}[\alpha, \beta] \), and (b) **Gaussian noise** adds Gaussian random variable, i.e., \( s' = s + z \), where \( z \sim \mathcal{N}(0, I) \). Here, \( s', s \) and \( z \) are all vectors (see Appendix J for more details). Similar to image inputs, augmentations are applied randomly across the batch.

- Best augmentations / invalid augmentations

  ![Augmentation Table]

  Automatic Data Augmentation for Generalization in Deep RL

  ^ augmentations are used naively. Lots of others work if you’re careful
Links

CURL: Contrastive Unsupervised Representations for Reinforcement Learning

Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels

Reinforcement Learning with Augmented Data

Automatic Data Augmentation for Generalization in Deep Reinforcement Learning

Berkeley’s summary of their papers

Relevant Reddit thread