RQDIA: REGULARIZING Q-VALUE DISTRIBUTIONS WITH IMAGE AUGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

rQdia (pronounced "Arcadia") regularizes Q-value distributions with augmented images in pixel-based deep reinforcement learning. With a simple auxiliary loss, that equalizes these distributions via MSE, rQdia boosts DrQ and SAC on $9/12$ and $10/12$ tasks respectively in the MuJoCo Continuous Control Suite from pixels, and Data-Efficient Rainbow on $18/26$ Atari Arcade environments. Gains are measured in both sample efficiency and longer-term training. Moreover, the addition of rQdia finally propels model-free continuous control from pixels over the state encoding baseline.

1 INTRODUCTION

Human perception is invariant to and remarkably robust against many perturbations, like discoloration, obfuscation, and low exposure. On the other hand, artificial neural networks do not intrinsically carry these invariance properties, though some invariances may be induced architecturally through inductive biases like convolution, kernel rotation, and dilation. In deep reinforcement learning (RL) from pixels, an agent is tasked to learn from raw pixels and must therefore learn to visually interpret a scene. Thus, recent approaches in deep RL have turned to the self-supervision and data augmentation techniques found in computer vision. Indeed, such contrastive learning auxiliary losses (Srinivas et al., 2020) or data augmentation regularizers (Yarats et al., 2021) have afforded greater sample efficiency and final scores in both the DeepMind Continuous Control Suite (Tassa et al., 2018) from pixels and Atari Arcade Learning Environments (Bellemare et al., 2013).

Nevertheless, pixel-based approaches continue to lag behind models that learn directly from state embeddings, not just in terms of sample efficiency, but also in final reward performance. For example, the recent DrQ (Yarats et al., 2021), an image augmentation-based regularizer to SAC (Haarnoja et al., 2018), reports falling 14.5% short of its state embedding-based SAC counterpart on the Cheetah Walk task. Such discrepancies indicate that visual representations are not yet up to par with state embeddings, at least not for locomotive continuous control. State embeddings have many properties that facilitate generalization, such as location invariance, and to a degree, invariance between morphological relations. If a subset of the dimensions of a state embedding always indicates feet position, then the relation of "one foot in front of the other" will be represented by those dimensions invariant to the placement of the robot's arms, head, or other body parts. Parametric visual encodings are not guaranteed to learn such invariances with respect to the robot's morphology.

This begs the question of what other signals are available in deep reinforcement learning which can guide visual representation learning towards more invariant encodings. We propose that Q-value distributions, the expected cumulative discounted reward across *distributions* of actions for a given state, offer such a signal. For a discrete set of actions $a_0, \ldots, a_n$, a state s_t, and a Q-function Q, a Q-value distribution is simply:

$$Q(s_t, a_0), \ldots, Q(s_t, a_n).$$

This distribution over actions for a state is a measure of the current *and* future value of each action for that state. We review states, actions, rewards, Q-values, and the reinforcement learning Markov Decision Process formal framework in the subsequent Section 2.1. This distribution carries many of the same invariances afforded by state embeddings, if not some additional ones. For example, if the optimal action is to put "one foot in front of the other," then the Q-value distribution reflects the

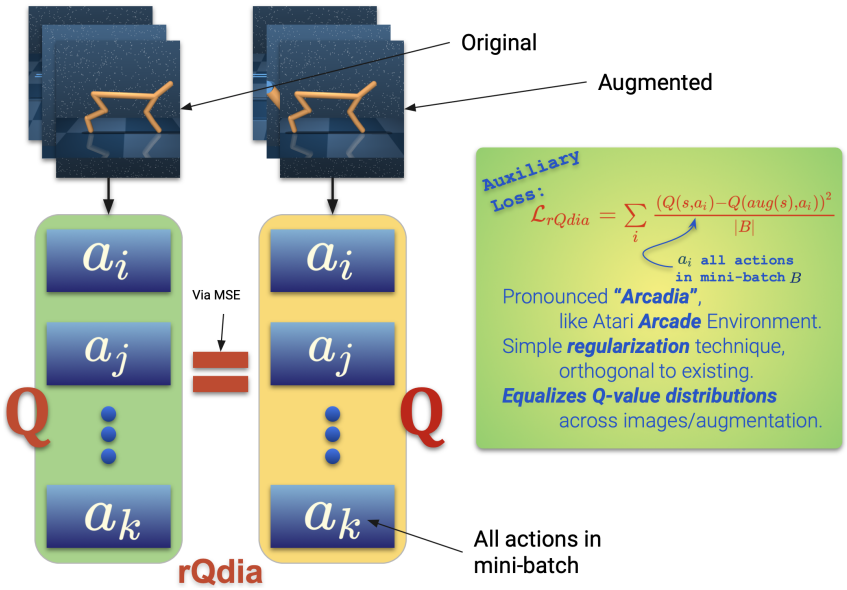

Figure 1: ***"rQdia in a nutshell"*** rQdia regularizes Q-value distributions across mini-batches of actions between images and their augmentations (*e.g.*, *crop/translate*) with a simple auxiliary loss.

preferability of this action over other actions regardless of (invariant to) where the agent is located, how it is holding its hands, or any other potentially superfluous contextual information.

While recent work has found that individual Q-values benefit from regularizing across augmented images (Yarats et al., 2021), we consider that Q-value *distributions* are also important, as they contain information not just about one action in isolation, but multiple actions in relation to one another.

2 Background

2.1 Deep Reinforcement Learning From Pixels

A Markov Decision Process (MDP) consists of an action $a \in \mathbb{R}^{d_a}$, state $s \in \mathbb{R}^{d_s}$, and reward $r \in \mathbb{R}$. "From pixels" assumes that state s is an image frame or multiple image frames stacked together. The action is sampled from a policy at any t time step $a_t \sim \pi(s_t)$. Taking such actions yields a trajectory $\tau = (s_0, a_0, s_1, a_1, ..., s_T)$ via the dynamics model $s_{t+1} \sim f(s_t, a_t)$ of the environment and its rewards $r_{t+1} = R(s_t, a_t)$. The goal is to maximize cumulative discounted reward $R(\tau) = \sum_{t=0}^{T} r_t \gamma^t$ where γ is the temporal discount factor. The optimal action for a state $a^*(s) = argmax_a Q^*(s, a)$ thus depends on the state-action value function, also known as the Q-value, $Q^\pi(s, a) = \mathbf{E}_{\tau \sim \pi}[R(\tau)|s_0 = s, a_0 = a]$.

2.2 Soft Actor-Critic

Soft-Actor Critic (SAC) (Haarnoja et al., 2018) is an RL algorithm which learns a Q-value function Q_θ parameterized by θ, optimized with one-step Bellman error, and a stochastic Gaussian policy $\pi_{\theta'}$, optimized by maximizing Q_θ and entropy, where θ' is the parameter subspace of θ corresponding to the policy network. In SAC, this Q-value function also includes entropy $H(s_t) = -\pi_{\theta'}(s_t) log(\pi_{\theta'}(s_t))$ as part of the reward. In practice, Q_θ and π_θ share a convolutional neural net encoder, and policy head π_θ is made differentiable via the reparameterization trick.

2.3 Rainbow

Rainbow (Hessel et al., 2018) similarly represents pixel-level inputs with a convolutional neural net, but operates on a discrete action space and thus directly estimates Q-values for each discrete action. To further improve performance, several refinements are used: these Q-values are sampled from a

multivariate Gaussian probabilistically rather than deterministically (Dabney et al., 2018), double Q networks (Van Hasselt et al., 2016), noise is injected into the network parameters (Plappert et al., 2017), dueling DQNs (Wang et al., 2016), n-step returns (Watkins, 1989), and mini-batches are sampled from a prioritized experience replay (Schaul et al., 2015).

3 Related Work

3.1 Continuous Control From Images

Continuous control from images is highly challenging as images are high-dimensional and supervisory signals in reinforcement learning (RL) are limited to scalar rewards. One way to mitigate the problem is to learn the dynamics of the environment with model-based algorithms then perform policy search through the learned model (Hafner et al., 2019b; Lee et al., 2019; Hafner et al., 2019a).

More recently, model-free methods like SAC with convolutional encoder received an enhancement by introducing an auxiliary image reconstruction task (Yarats et al., 2019; Srinivas et al., 2020; Sermanet et al., 2018; Dwibedi et al., 2018), which provides additional self-supervision for visual representation learning.

3.2 Image Augmentation

Image augmentation is originally developed to counter over-fitting in computer vision, which applies transformations that include color shift, affinity translation, scale, etc. (Ciregan et al., 2012; Cireşan et al., 2011; Simard et al., 2003; Krizhevsky et al., 2012b; Chen et al., 2020), to the input image where the output label is invariant. In this way, the neural network's search space is restricted to converging to more invariant solutions during optimization, thus benefiting its generalizability. Such techniques have become largely standard in computer vision since AlexNet (Krizhevsky et al., 2012a). A similar idea is borrowed in RL for continuous control from images (Yarats et al., 2021; Srinivas et al., 2020) and discrete control Atari Schwarzer et al. (2020). However, the reward is not invariant to all transformations. For example, if the agent is rewarded for moving right, applying a rotation to the image may not preserve the reward. In Yarats et al. (2021), the author investigated several common image transformations and conclude that random shifts strike a good balance between simplicity and performance.

3.3 Contrastive Learning

Furthermore, contrastive learning aims at grouping similar samples closer and diverse samples farther from each other, which will yield more invariant visual representations that benefit recognition (Chen et al., 2020; He et al., 2020; Misra & Maaten, 2020; Henaff, 2020). Usually, this is done by adding auxiliary losses to classify associated pairs created by image augmentation positively while discriminating random pairs negatively. In the RL domain, CURL (Srinivas et al., 2020) acquired higher performance on continuous control tasks from pixels by employing such an auxiliary loss in a SAC architecture.

3.4 Regularization

Many simple regularizers have benefited generalization in deep RL, including dropout (Srivastava et al., 2014), L2 (Cobbe et al., 2019), and most recently and commonly, entropy (Ziebart et al., 2008; Haarnoja et al., 2018; Nachum et al., 2017; Williams & Peng, 1991). Very recently, image augmentation has been used to regularize Q-values in continuous control (Yarats et al., 2021). However, these regularizations are fully complementary to rQdia. rQdia regularizes Q-value distributions rather than individual Q-values, and can be incorporated with a simple auxiliary loss rather than averaging Q-value predictions in the network inference itself.

Simple random translations have been found to be the most useful in the DMControl from pixels setting (Srinivas et al., 2020; Yarats et al., 2021). A variety of different augmentations are useful for different games in the Procgen benchmark (Raileanu et al., 2020), and Yarats et al. (2021) uses Intensity variation for the Atari environments.

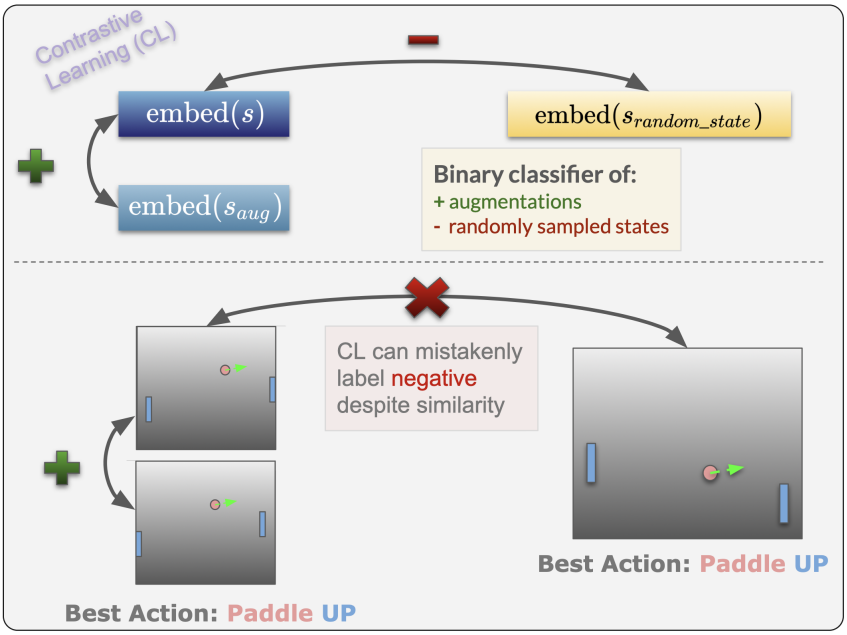

Figure 2: Relying on contrastive learning has disadvantages that this emerging family of approaches based on simple equalization does not.

3.5 Generalization And Sample Efficiency

Scene variation has been used to improve generalization in RL, such as in robotics Gupta et al. (2018), Meta-World (Yu et al., 2020), and ProcGen (Mohanty et al., 2021). Domain randomization (Tobin et al., 2017; Slaoui et al., 2019) similarly varies the scene except by modifying the dynamics of the environment simulator itself.

4 Methods

As mentioned previously, we use Q-value distributions to lend a reinforcement learner greater visual invariance. Now, recently DrQ used image augmentation to induce greater invariance. The augmentation used was a simple padding and crop, *e.g.*, a translation.

Unsurprisingly, given what is known about data augmentation in computer vision, the addition of this small translation invariance helped visual representation learning. Interestingly, this benefit far exceeded that of the preceding contrastive learning based approach, where the augmented image was correlated with the current state, and randomly sampled historical images were "contrasted." This method, CURL (Srinivas et al., 2020), successfully adapted SAC (Haarnoja et al., 2018) to be compatible with pixel-level inputs instead of state embeddings. However, DrQ's direct use of the augmented image to predict Q-values outperformed the more conceptually complex contrastive learning approach. This is perhaps due to the greater freedom afforded to the model to map pixels to a latent space that corresponds directly with the end goal of evaluating the quality of an action at a given state, rather than inducing a decoupled latent space similarity with contrastive learning. Moreover, however, this direct use of the augmented image circumvents a major downside of naive contrastive learning, which is that negative samples are "contrasted" despite being sampled randomly from the agent's state history. This means that policy-relevant states may be de-associated despite the policy relevance. Please see Figure 2 for an example of this in the Atari classic, Pong.

DrQ does not require an auxiliary loss, but rather just averages the Q-values of the anchor image with that of the augmented image, effectively training both in tandem.

This is however optimized for a single Q-value, which is the one associated with the selected action. This is also the only Q-value for which we have a label, since we do not know the ground truth Q-value of an action that was not taken.

Furthermore, we do not have a finite number of Q-values in environments such as the DeepMind Continuous Control Suite, where locomotive motor control is unrestricted spatially per joint and thus infinite.

Our solution in a nutshell: equalize Q-value distributions between anchors and augmentations by convolving all actions in the mini-batch. Why the mini-batch instead of randomly sampling actions? Simply because the mini-batch is already conveniently available during training and equates to using randomly sampled historical actions. By using historical actions, we can actually compute Q-value distributions with moderately realistic state-action pairs that could feasibly be encountered in simulation. We do not employ a heuristic for sampling such actions here in this work, but we note that sampling actions based on a more sophisticated measure, such as state similarity, is also possible.

Like DrQ, we use random translation as our augmentation technique. Specifically, we pad by 4 pixels, then crop randomly inward by 4 pixels. Without loss of generality, let's refer to this augmentation function as $aug(\cdot)$.

Further, let's define our state s_t and batch size n. Then rQdia is simply:

$$\mathcal{L}_{rQdia} = \frac{1}{n} \sum_{i<n} (Q(s_t, a_i) - Q(aug(s_t), a_i))^2, \tag{1}$$

where $a_0, ..., a_{n-1}$ are the set of actions in the mini-batch.

Then this auxiliary loss is simply added to the RL agent's standard loss term. Voila, rQdia (summarized "in a nutshell" in Figure 1). This is applied in parallel for each of the states in the mini-batch. If mini-batches are very large, it is possible to merely convolve a subset of the n actions in the mini-batch for the a_i in Equation 2.

Outside of continuous control, we just use the full discrete action space rather than convolving mini-batch-sampled actions. In Rainbow in particular, which uses distributional probabilistic Q-values, we also found that mean-squared error, as above in Equation 2, between Q-value logits, performed best, as opposed to using KL-divergence between their associated distributional probabilities, which would have been an alternative way to minimize the difference between the two Q-value distributions in Rainbow.

Pseudo-code for adding rQdia to continuous control models like SAC-AE and DrQ is shown in Algorithm 1. All code for continuous *and* discrete action spaces will be released.

5 Experiments

5.1 100k Continuous Control

rQdia added to DrQ yields several noteworthy results. Firstly, we find sample efficiency greatly improves with the addition of rQdia. In the 100k DeepMind Control Suite setting, rQdia benefits or improves over recent algorithms on 5/6 environments, as shown in Table 1.

Interestingly, rQdia hurts performance on the finger spin environment when combined with DrQ, though improves over it when combined with SAC-AE as we will see later. We are not sure why this regularization cuts performance here, while it benefits performance in the more complex tasks or when combined with the older SAC-AE. Perhaps there is a tradeoff between generality and in-

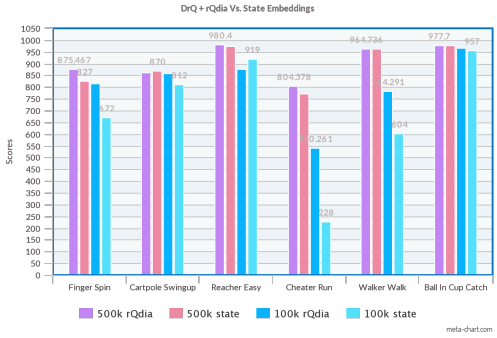

Figure 3: DrQ + rQdia pushes DrQ over the top of the state embedding baseline.

Algorithm 1 rQdia added to original Soft Actor-Critic algorithm, pseudocode courtesy of Achiam (2018). SAC is a good base framework and can be easily expanded to DrQ (Yarats et al., 2021).

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
Denote augmentation function $aug(\cdot)$
repeat
  Observe state s and select action $a \sim \pi_\theta(\cdot|s)$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets for the Q functions:

$$y(r, s', d) = r + \gamma(1 - d)\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} \left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2 \qquad \text{for } i = 1, 2$$

      Update rQdia by one step of gradient descent using:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,\ldots),(s_j,a,\ldots)\in B} \left(\min_{i=1,2} Q_{\phi_i}(s, a) - \min_{i=1,2} Q_{\phi_i}(aug(s), a)\right)^2$$

      Update policy by one step of gradient ascent using:

$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s)|\, s\right)\right),$$

      where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt θ via the reparametrization trick.
      Update target networks with:

$$\phi_{\text{targ},i} \leftarrow \rho\phi_{\text{targ},i} + (1 - \rho)\phi_i \qquad \text{for } i = 1, 2$$

    end for
  end if
until convergence

domain performance which would be observed if the training and testing settings were different or as difficulty increases. This could occur if the model overfits to an environment and relies on disagreeing/biased Q-value distributions.

Amazingly, in this 100k setting, rQdia substantially outperformed SAC from state embeddings. That is to say, that training from pixels actually learns *faster* than training from state embeddings with rQdia. This is less often the case with other models (5/6 for DrQ + rQdia, 2/6 for DrQ with 1 approximately the same). We speculate that this is because rQdia induces original invariances early on in training that state embeddings do not possess, thanks to the added Q-value distribution invariance across image translatons. DrQ + rQdia is compared to state embeddings in Figure 3

Table 1: 500k and 100k scores in the DeepMind Continuous Control Suite. Minimal bound performances are shown. Due to computational limits, rQdia was run with a batch size of 128 and we compare against the results reported in Yarats et al. (2021), where a batch size of 512 was used. They found bigger batch sizes correspond to better performance, making our model disadvantaged. rQdia is augmented to DrQ with a reduced batch size of 128. We report our best mean episode reward averaged across 3 random seeds. We could not use more random seeds since the MuJoCo license is restricted to a single machine and more licenses are prohibitively expensive. Despite these disadvantages, rQdia generally benefits compared to computationally bigger state of the art models. We see especially prominent gains in sample efficiency, as shown by the 100k step setting, and in the arguably two most challenging environments, Cheetah and Walker. SAC State is SAC trained from state embeddings, which amazingly rQdia often also beats in performance or comes close to.

	From Pixels					**State Emb**
500k Step Scores	DrQ + rQdia (Ours)	DrQ	CURL	PlaNet	SAC-AE	SAC State
Finger Spin	875.467	**938**	874	718	914	927
Cartpole Swingup	862.128	**868**	861	787	730	**870**
Reacher Easy	**980.4**	942	904	588	601	975
Cheetah Run	**804.378**	660	500	568	544	772
Walker Walk	**964.736**	921	906	478	858	964
Ball In Cup Catch	**977.7**	963	958	939	810	**979**
100k Step Scores						
Finger Spin	816.367	**901**	779	560	747	672
Cartpole Swingup	**857.932**	759	592	563	276	812
Reacher Easy	878.15	601	517	82	225	**919**
Cheetah Run	**540.261**	344	307	165	252	228
Walker Walk	**784.291**	612	344	221	395	604
Ball In Cup Catch	**967.233**	913	772	710	338	957

5.2 500K CONTINUOUS CONTROL

In the 500k continuous control setting, rQdia benefited training on 4/6 environments, as seen in Table 1. In those environments, we also saw performance roughly on par with or better than SAC from state embeddings on all 4/6 of these, compared to 1/6 for DrQ without rQdia. In both settings, rQdia combined with DrQ surpasses SAC from state embeddings more frequently and by wider margins generally than either DrQ (Yarats et al., 2021) or CURL (Srinivas et al., 2020) standalone. In the Appendix, we provide full graphs of these runs.

5.3 ATARI ARCADE 100K

In the standard 100k efficient Atari benchmark, as reported by CURL (Srinivas et al., 2020), the addition of rQdia to Data-Efficient Rainbow (van Hasselt et al., 2019) performs well compared to CURL, which relied on contrastive learning. We notice some remarkable improvements over both CURL and Eff. Rainbow on 15/26 environments, with an especially large gain for Frostbite, a game that depends heavily on translation invariance due to its visual symmetries.

We note that since Rainbow's Q-value distribution is equivalent to its action distribution, that rQdia regularizes the action distribution directly here, while only doing so indirectly via Q-value distributions in continuous control. It is a question for future work whether regularizing actions directly in continuous control (equalizing their log probabilities) would benefit performance there.

We note that rQdia is orthogonal/complementary to CURL, DrQ, and other such methods, though we did not find performance improvements from rQdia in combination with CURL like we did with DrQ. DrQ has not yet released their Atari DQN code to the public.

Table 2: 100k steps in the Atari Arcade Learning Environment. rQdia and CURL are both built on top of Data-Efficient Rainbow (van Hasselt et al., 2019). We report the best mean episode reward averaged across 3 random seeds, which is 2 seeds fewer than reported by DrQ (Yarats et al., 2021). We note that DrQ did not release their Atari code at time of submission. rQdia improves upon Data-Efficient Rainbow on 24/26 environments, upon CURL on 15/26 environments, and achieves higher mean episode rewards and normalized scores. We reproduced CURL with the default hyperparams of the code released by Srinivas et al. (2020), using the same hyperparams for rQdia+Eff. Rainbow as well. Scores for Eff. Rainbow, human, and random are copied over from those reported in DrQ (Yarats et al., 2021). rQdia improves Eff. Rainbow by larger margins than CURL.

Atari Arcade Environments	rQdia-Rainbow	CURL	Eff. Rainbow	Human	Random
Alien	**1001.00**	960.00	739.9	7127.7	227.8
Amidar	185.40	**219.00**	188.6	1719.5	5.8
Assault	**469.35**	417.90	431.2	742.0	222.4
Asterix	**817.50**	641.67	470.8	8503.3	210.0
BankHeist	**136.50**	24.33	51.0	753.1	14.2
BattleZone	**14700.00**	7700.00	10124.6	37187.5	2360.0
Boxing	**6.50**	3.40	0.2	12.1	0.1
Breakout	**5.35**	3.33	1.9	30.5	1.7
ChopperCommand	880.00	**913.33**	861.8	7387.8	811.0
CrazyClimber	**18083.33**	4070.00	16185.3	35829.4	10780.5
DemonAttack	910.00	**1213.00**	508.0	1971.0	152.1
Freeway	24.53	**28.90**	27.9	29.6	0.0
Frostbite	**2613.00**	444.00	866.8	4334.7	65.2
Gopher	**446.67**	398.00	349.5	2412.5	257.6
Hero	**8539.50**	7079.00	6857.0	30826.4	1027.0
Jamesbond	285.00	**355.00**	301.6	302.8	29.0
Kangaroo	**880.00**	**880.00**	779.3	3035.0	52.0
Krull	2116.00	**3944.00**	2851.5	2665.5	1598.0
KungFuMaster	**9620.00**	6850.00	14246.1	22736.3	258.5
MsPacman	1363.33	**1484.00**	1204.1	6951.6	307.3
Pong	**-18.33**	-19.60	19.3	14.6	-20.7
PrivateEye	0.00	**100.00**	97.8	69571.3	24.9
Qbert	**1489.17**	890.00	1152.9	13455.0	163.9
RoadRunner	8180.00	1510.00	**9600.0**	7845.0	11.5
Seaquest	**349.33**	251.00	**354.1**	42054.7	68.4
UpNDown	1524.67	**3590.67**	2877.4	11693.2	533.4
Better Mean Episode Reward	**16/26**	10/26	2/26	-	-
Mean Human-Normalized Score	**17.36%**	7.76%	16.27%	100%	0%
Med Human-Normalized Score	**17.83%**	13.37%	13.99%	100%	0%

5.4 RQDIA-SAC

We also evaluated how rQdia works independent of DrQ when augmented by itself as an auxiliary loss to SAC-AE. Indeed rQdia as a standalone auxiliary loss performs well, often even better than either DrQ and CURL. Please see Figure 3.

While we note that rQdia is orthogonal implementation-wise to CURL, we found that combining the two did not benefit performance. Similarly, combining CURL with DrQ tended to hurt performance. rQdia and DrQ on the other hand, complemented each other well.

6 DISCUSSION

6.1 LIMITATIONS

One limitation of rQdia is that it assumes a benefit to a certain augmentation invariance. Translation invariance for example might not be as useful in environments where most objects are held within a consistent axis.

Table 3: Comparing rQdia added to just SAC-AE to state of the art models that succeeded SAC-AE. Similarly to Table 1, 500k and 100k performances are shown, with our model disadvantaged in terms of batch size, and 3 random seeds. To our surprise, results are largely on par with DrQ + rQdia, occasionally exceeding it, and other state of arts.

	From Pixels					**State Emb**
500k Step Scores	SAC-AE + rQdia (Ours)	DrQ	CURL	PlaNet	SAC-AE	SAC State
Finger Spin	**998**	938	874	718	914	927
Cartpole Swingup	863.491	**868**	861	787	730	**870**
Reacher Easy	**989.1**	942	904	588	601	975
Cheetah Run	**804.546**	660	500	568	544	772
Walker Walk	492.586	921	906	478	858	**964**
Ball In Cup Catch	**975**	963	958	939	810	**979**
100k Step Scores						
Finger Spin	**948**	901	779	560	747	672
Cartpole Swingup	**858.291**	759	592	563	276	812
Reacher Easy	868.5	601	517	82	225	**919**
Cheetah Run	**553.921**	344	307	165	252	228
Walker Walk	411.897	**612**	344	221	395	604
Ball In Cup Catch	947.7	913	772	710	338	**957**

Moreover, rQdia ensures consistency between Q-value distributions across such perturbations, which means that models that do not need to learn such visual augmentations are expected to learn a more complex Q function.

On the other hand, such environments where these invariances are not useful or important may leverage rQdia to learn more invariant representations that could potentially better generalize to more complex environments.

6.2 Ethics

rQdia is a simple regularizer that contributes to the generalization of deep reinforcement learning models. While we hope deep RL continues to improve and its applications and abilities expand, we would be remiss not to note the destructive potential of the field, ranging from autonomous weapons to economic exploitation. However, we are optimistic that RL can do much more good than bad for society. Autonomous agents that can interact with the world and streamline robotics opens the door for countless medical, social, and economic benefits as well.

6.3 Reproducability

We have submitted Code together with the paper, a full pseudocode is offered in Algorithm 1 for reproducing continuous control, and a relevant snippet of Code is shared in Appendix C for reproducing Atari.

7 Conclusion

We presented a simple regularizer for model-free reinforcement learning that may easily be integrated into existing reinforcement learning frameworks. With the inclusion of this auxiliary loss, we attain strong performance compared to baseline models, including recent state of the arts. By regularizing Q-value distributions, we further enforce the invariances afforded by image augmentation techniques such that Q-value distributions are preserved under these perturbations. Consequently, we observe improvements in sample efficiency and final reward in the DeepMind Continuous Control Suite and environments in the Atari Arcade Learning Environments.

REFERENCES

Joshua Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3642–3649. IEEE, 2012.

Dan C Cireşan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*, 2011.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 1282–1289. PMLR, 2019.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Debidatta Dwibedi, Jonathan Tompson, Corey Lynch, and Pierre Sermanet. Learning actionable representations from visual observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1577–1584. IEEE, 2018.

Abhinav Gupta, Adithyavairavan Murali, Dhiraj Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *arXiv preprint arXiv:1807.07049*, 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019b.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pp. 4182–4192. PMLR, 2020.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012a.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012b.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6707–6717, 2020.

Sharada Mohanty, Jyotish Poonganam, Adrien Gaidon, Andrey Kolobov, Blake Wulfe, Dipam Chakraborty, Gražvydas Šemetulskis, João Schapke, Jonas Kubilius, Jurgis Pašukonis, et al. Measuring sample efficiency and generalization in reinforcement learning benchmarks: Neurips 2020 procgen benchmark. *arXiv preprint arXiv:2103.15332*, 2021.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *arXiv preprint arXiv:1702.08892*, 2017.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with momentum predictive representations. *CoRR*, abs/2007.05929, 2020. URL `https://arxiv.org/abs/2007.05929`.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3. Citeseer, 2003.

Reda Bahi Slaoui, William R Clements, Jakob N Foerster, and Sébastien Toth. Robust visual domain randomization for reinforcement learning. *arXiv preprint arXiv:1910.10537*, 2019.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. CURL: contrastive unsupervised representations for reinforcement learning. *CoRR*, abs/2004.04136, 2020. URL `https://arxiv.org/abs/2004.04136`.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Hado van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *arXiv preprint arXiv:1906.05243*, 2019.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=GY6-6sTvGaf`.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A ARCHITECTURE

The SAC-AE architecture we use is the same as [38], consisting of a shared encoder and distinct policy and Q-function heads. The CNN encoder is shared by the actor and critic; the critic consists of two ReLU-activated 3-layer MLP Q-networks; and the actor is a single ReLU-activated 3-layer MLP Gaussian policy head. We modify the code provided by [30]: `https://github.com/MishaLaskin/curl`.

Atari environments were tested with a Rainbow architecture inspired by [35] and built on the variant implemented in tandem with CURL in [30]. We added the rQdia auxiliary loss to their code sans the CURL-related portion. This code may be found here: `https://github.com/aravindsrinivas/curl_rainbow`.

B TRAINING

All hyperparameters were preserved from the original implementations discussed above. They are reviewed in Tables 4 and 5, except for the substitution of batch size since we used a batch size of 128 while [30, 37] used 512, giving those models a reasonable advantage.

C CODE

```
rQdia_loss = torch.nn.functional.mse_loss(log_aug_dist, log_anchor_dist)
loss = loss + rQdia_loss
```

Figure 4: Pytorch code for rQdia in Rainbow Atari.

Code for continuous control and discrete Atari will be released on GitHub upon notification of decision and is provided separately together with the supplementary material.

The rQdia loss in Rainbow is a simple mean squared error between the anchor and augmentation's respective Q-value distributions, as shown in Figure 4.

Continuous control is more complicated.

Param	Value
Observation Size	(84, 84)
Replay Buffer Size	100000
Initial Steps	1000
Stacked Frames	3
Action Repeat	2 finger, spin;
walker, walk	
8 cartpole, swingup	
4 otherwise	
MLP Hidden Units	1024
Evaluation Episodes	10
Optimizer	ADAM
Learning Rate $(f_\theta, \pi_\psi, Q_\phi)$	0.001
Learning Rate (α)	0.0001
Batch Size	128
Q Function EMA τ	0.01
Critic Target Update Freq	2
Conv Layers	4
Number of Filters	32
Non-Linearity	ReLU
Encoder EMA τ	0.05
Latent Dimension	50
Discount γ	0.99
Initial Temperature	0.1

Table 4: Hyperparameters for rQdia-DrQ

Param	Value
Observation Size	(84, 84)
Replay Buffer Size	100000
Frame Skip	4
Action repeat	4
Q-network Channels	32, 64
Q-network Filter Size	$5 \times 5, 5 \times 5$
Q-network Stride	5, 5
Q-network Hidden Units	256
Momentum τ	0.001
Non-Linearity	ReLU
Reward Clipping	$[-1, 1]$
Multi Step Return	20
Min replay size	
for sampling	1600
Max Frames Per Episode	108K
Target network	
update period	2000 updates
Support Of Q-dist	51 bins
Discount γ	0.99
Batch Size	32
Optimizer	ADAM
Learning Rate	0.9
(β_1, β_2)	(0.9, 0.999)
Optimizer ϵ	0.000015
Max Grad Norm	10
Noisy Nets Parameter	0.1
Priority Exponent	0.5
Priority Correction	$0.4 \rightarrow 1$

Table 5: Hyperparameters for rQdia-Rainbow

D CONTINUOUS CONTROL

Plots of continuous control are shown in Figure 5.

E ATARI ARCADE

Plots for Atari Arcade are shown in Figure 6.

F KL-DIVERGENCE LOSS FOR RQDIA-RAINBOW

We also evaluate rQdia as a KL divergence loss between the two distributional Q-value predictions yielded by Rainbow:

$$\mathcal{L}_{rQdia} = D_{KL}(Q(s_t, a_i)||Q(aug(s_t), a_i)). \tag{2}$$

While performance is not as good on all measures, as shown in Table 6, KL divergence does benefit mean human-normalized score and yields significant improvements on particular environments; most notably, rQdia-D_{KL} attains a score of 3655.80 on PrivateEye, compared to 0 for rQdia-MSE, 100 for CURL, and 69571.3 for a human.

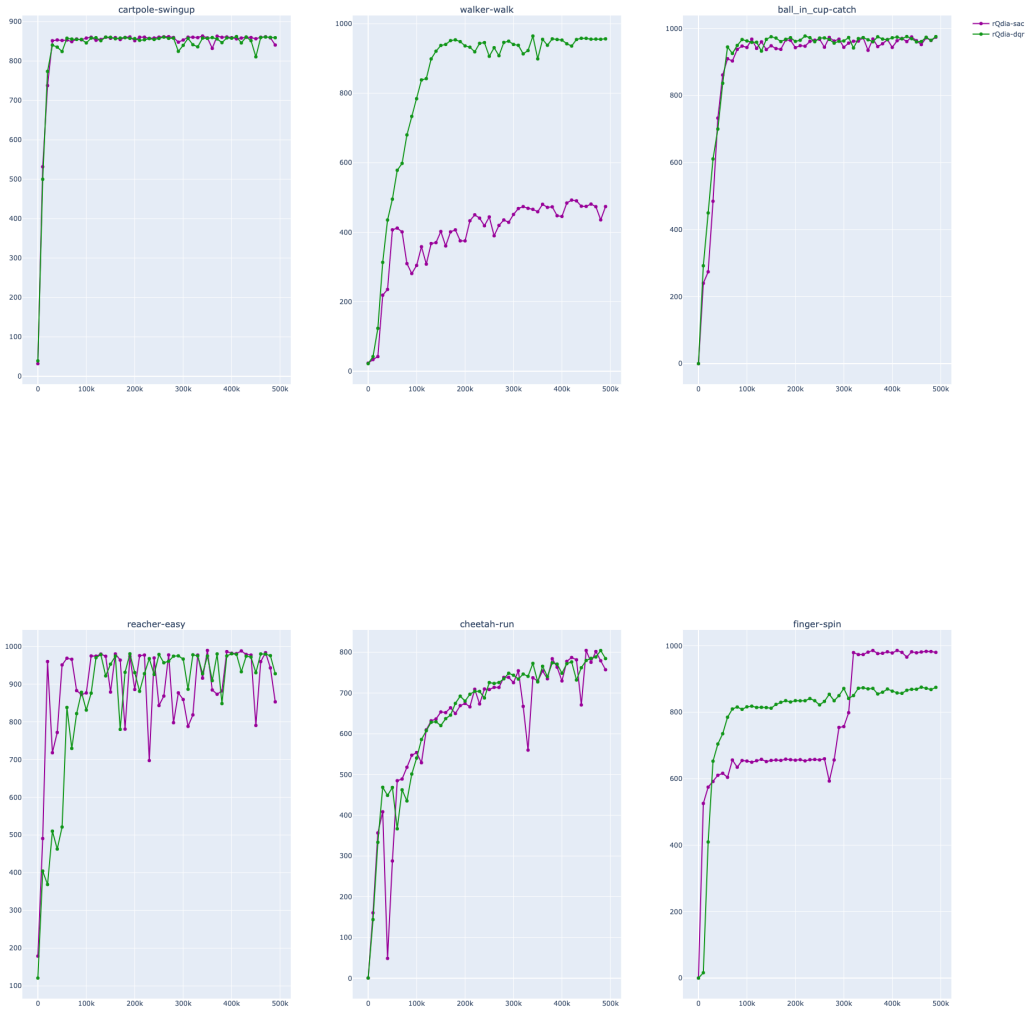

Figure 5: Continuous control tasks.

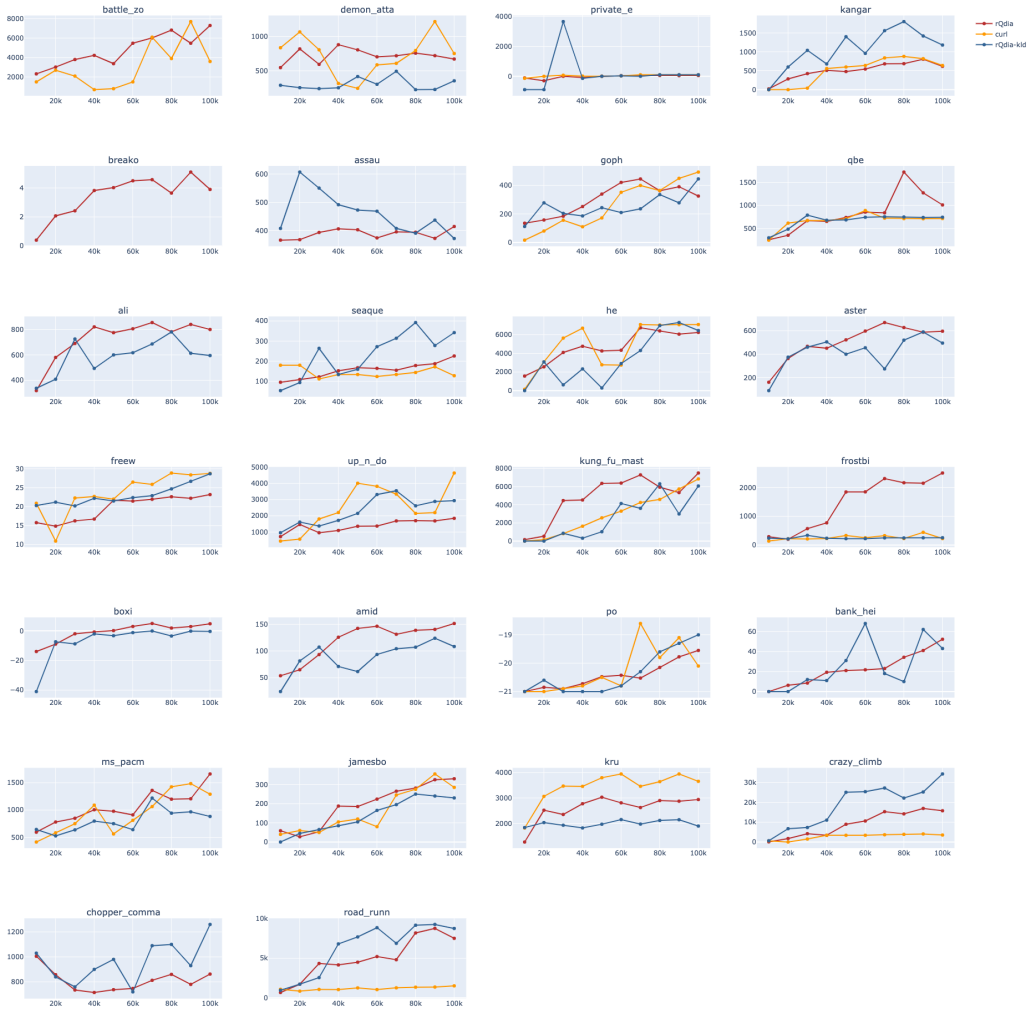

Figure 6: Atari Arcade tasks.

Table 6: KL divergence interestingly benefits mean human-normalized score, but is less optimal for other measures compared to mean squared error.

Atari Arcade Environments	rQdia-D_{KL}	rQdia-MSE	CURL-Rainbow	Human	Random
Alien	780.00	**1001.00**	960.00	7127.7	227.8
Amidar	123.90	185.40	**219.00**	1719.5	5.8
Assault	**606.90**	469.35	417.90	742.0	222.4
Asterix	590.00	**817.50**	641.67	8503.3	210.0
BankHeist	68.00	**136.50**	24.33	753.1	14.2
BattleZone	7900.00	**14700.00**	7700.00	37187.5	2360.0
Boxing	-0.10	**6.50**	3.40	12.1	0.1
Breakout	4.60	**5.35**	3.33	30.5	1.7
ChopperCommand	**1260.00**	880.00	913.33	7387.8	811.0
CrazyClimber	**34280.00**	18083.33	4070.00	35829.4	10780.5
DemonAttack	494.50	910.00	**1213.00**	1971.0	152.1
Freeway	28.70	24.53	**28.90**	29.6	0.0
Frostbite	340.00	**2613.00**	444.00	4334.7	65.2
Gopher	446.00	**446.67**	398.00	2412.5	257.6
Hero	7290.00	**8539.50**	7079.00	30826.4	1027.0
Jamesbond	250.00	285.00	**355.00**	302.8	29.0
Kangaroo	**1800.00**	880.00	880.00	3035.0	52.0
Krull	2154.00	2116.00	**3944.00**	2665.5	1598.0
KungFuMaster	6310.00	**9620.00**	6850.00	22736.3	258.5
MsPacman	1219.00	1363.33	**1484.00**	6951.6	307.3
Pong	-19.00	**-18.33**	-19.60	14.6	-20.7
PrivateEye	**3655.80**	0.00	100.00	69571.3	24.9
Qbert	790.00	**1489.17**	890.00	13455.0	163.9
RoadRunner	**9270.00**	8180.00	1510.00	7845.0	11.5
Seaquest	**392**	349.33	251.00	42054.7	68.4
UpNDown	3541	1524.67	**3590.67**	11693.2	533.4
Better Mean Episode Reward	7/26	**12/26**	7/26	-	-
Mean Human-Normalized Score	**20.83%**	17.36%	7.76%	100%	0%
Median Human-Normalized Score	11.31%	**17.83%**	13.37%	100%	0%