Improved Regularisation for Automatic Data Augmentation

Project proposal document

Sunday 13th February, 2022

Contents

Team Members:

- Hitul Desai, 18D070009
- Advait Kumar, 18D070003
- Aditya Badola, 190050006

1 Introduction

Learning from visual observations is a challenging problem in Reinforcement Learning. Although CNNs have proven to be quite successful, the method still fails on Data efficiency and Generalization to new environments. To this end, **RAD** [4] introduced a simple plug-and-play module which can be used to enhance most RL algorithms. They introduced data augmentations like color jitter, crop, translate, patch cut out and random convolutions.

However, this requires expert domain knowledge in terms of what transformations to choose when. **DrAC** [6] proposes two regularization terms for the policy and value function, required to make the use of data augmentation theoretically sound for actor-critic algorithms. [6] achieves state-of-the-art results on the OpenAI Procgen benchmarks and outperforms popular RL algorithms on DeepMind Control tasks. We will be looking at setting up a baseline of [6] on Procgen benchmarks ourselves and extend it further by using different algorithms for selecting the appropriate augmentation, and also practically experiment and find out best regularization functions for policy and value functions to optimize performance on the OpenAI Procgen benchmark.

2 Background and Terminology

2.1 Actor-Critic algorithm

An actor-critic algorithm is a **Temporal Difference** (TD) version of policy gradient. Essentially, the "critic" estimates the Q-value, and the "actor" updates the policy distribution using policy gradient based on the Q-value estimated by "critic".

2.2 PPO

Proximal Policy Iteration [7] is a policy optimization actor-critic algorithm that learns the policy π and value function V parametrised by θ to find an optimal policy. PPO samples the data and optimizes the objective function using stochastic gradient ascent *alternatively*. The objective function to be maximized at each iteration is given as:

$$J_{\text{PPO}} = L_t(\theta) = L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)$$

where:

- $L^{CLIP} = \hat{\mathbb{E}}_t \left[min \left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$
- $r_t(\theta)$ is the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ [8]
- $\hat{A}_t$ is the estimate of the advantage function at timestep t
- L_t^{VF} is squared-error loss $(V_\theta(s_t) - V_t^{targ})^2$
- $S[\pi_\theta]$ is the entropy bonus for aiding exploration

3 Data Augmentation in Reinforcement Learning

Image augmentation has been successfully applied in Computer Vision for improving generalization on object classification tasks. However, that is not always the case in reinforcement learning. Tasks in computer vision as well as some tasks in RL are invariant to image transforms, however in RL, that may not always be so. For example, the reward obtained for left-right actions taken in a game might be different if the image is flipped horizontally. Moreover, data augmentation cannot be directly used as it changes the objective function for [7]. Hence the paper by Raileanu et al. [6] proposes regularization factors to be introduced to policy and value functions.

4 Policy and Value Function Regularization

Raileanu et al. [6], inspired by Krosikov et al. [9] proposes regularization terms that constrains both the actor and the critic. This differs from Krosikov et al. [6] as it only applies a regularization to Q-function. This allows the method to be used with a different and larger class of algorithms, namely those which involve learning a policy and value function.

Raileanu et al. [6] defines an optimality state invariant transformation function $f : \mathcal{S} \times \mathcal{H} \to \mathcal{S}$ as a mapping that preserves both the agent's policy π and value function V such that $V(s) = V(f(s, \nu))$ and $\pi(a|s) = \pi(a|f(s, \nu)) \; \forall \; s \in S, \nu \in \mathcal{H}$, where ν is the parameter of $f(.)$, drawn from the set of all possible parameters $\mathcal{H}$.

To ensure that the policy and value functions are invariant to such transformation of the input state, the following were proposed by [6]:

- A loss term for regularizing the policy:

$$G_\pi = KL\left[\pi_\theta(a|s) | \pi_\theta(a|f(s, \nu))\right]$$

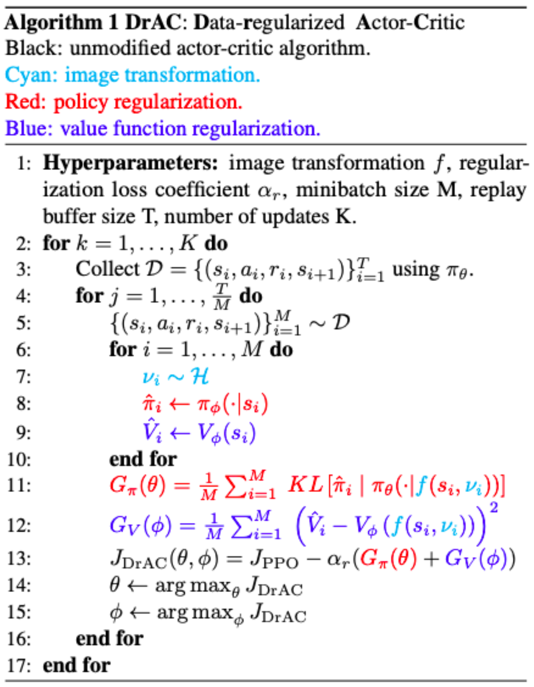
- An additional loss term for regularising the value function:

$$G_V = (V_\phi(s) - V_\phi(f(s, \nu)))^2$$

Thus, the new algorithm **DrAC** (Data regularization Actor Critic) maximizes the objective function:

$$J_{\text{DrAC}} = J_{\text{PPO}} - \alpha_r(G_\pi + G_V)$$

where α_r is the weight of the regularization term



Algorithm 1 DrAC: Data-regularized Actor-Critic
Black: unmodified actor-critic algorithm.
Cyan: image transformation.
Red: policy regularization.
Blue: value function regularization.

```
1: Hyperparameters: image transformation $f$, regularization loss coefficient $\alpha_r$, minibatch size M, replay buffer size T, number of updates K.
2: for $k = 1, \ldots, K$ do
3:     Collect $\mathcal{D} = \{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^{T}$ using $\pi_\theta$.
4:     for $j = 1, \ldots, \frac{T}{M}$ do
5:         $\{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^{M} \sim \mathcal{D}$
6:         for $i = 1, \ldots, M$ do
7:             $\nu_i \sim \mathcal{H}$
8:             $\hat{\pi}_i \leftarrow \pi_\phi(\cdot|s_i)$
9:             $\hat{V}_i \leftarrow V_\phi(s_i)$
10:        end for
11:        $G_\pi(\theta) = \frac{1}{M}\sum_{i=1}^{M} KL\left[\hat{\pi}_i \mid \pi_\theta(\cdot|f(s_i, \nu_i))\right]$
12:        $G_V(\phi) = \frac{1}{M}\sum_{i=1}^{M} \left(\hat{V}_i - V_\phi\left(f(s_i, \nu_i)\right)\right)^2$
13:        $J_{\text{DrAC}}(\theta, \phi) = J_{\text{PPO}} - \alpha_r(G_\pi(\theta) + G_V(\phi))$
14:        $\theta \leftarrow \arg\max_\theta J_{\text{DrAC}}$
15:        $\phi \leftarrow \arg\max_\phi J_{\text{DrAC}}$
16:    end for
17: end for
```

5 Automatic Data Augmentation

Since different tasks require different types of data augmentation we need a method that can effectively identify the transformations for the given task. Such a technique would significantly reduce the computational requirements for applying data augmentation in RL. The paper proposes three different algorithms for this task. However, because of time constraints, we will only be following the UCB-DrAC algorithm [6].

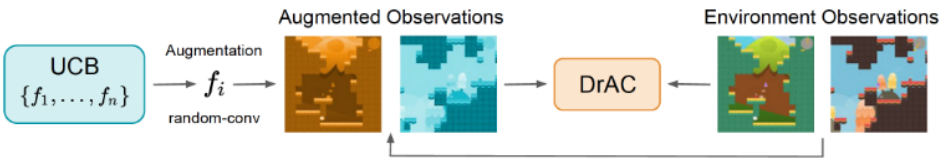


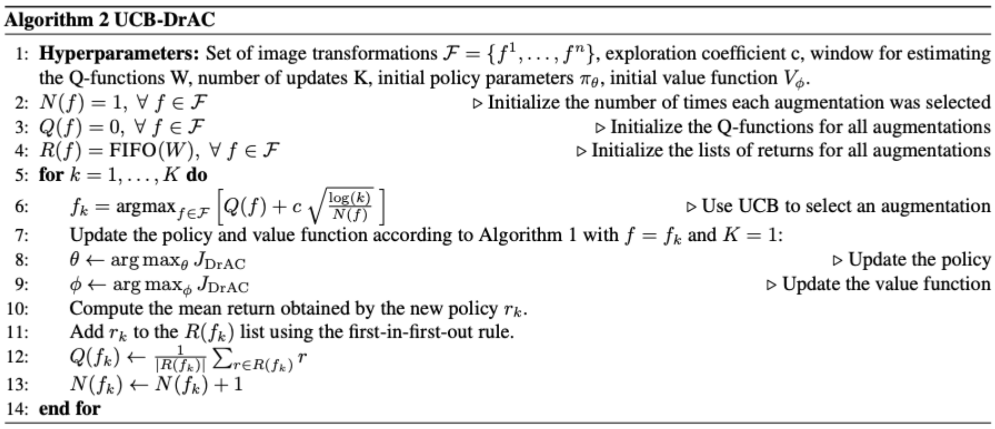
Figure 1: The augmented and original observations are passed to DrAC agent that learns a policy and value function invariant to this transformation, based on the observations

5.1 UCB-DrAC

The problem of selecting a data augmentation from a given set can be formulated as a multiarmed bandit problem, where the action space is the set of available transformations $F = \{f_1, ..., f_n\}$. A popular algorithm for such settings is the upper confidence bound or UCB which selects actions according to the following policy:

$$f_t = argmax_{f \in \mathcal{F}} \left[Q_t(f) + c\sqrt{\frac{log(t)}{N_t(f)}} \right]$$

Here f_t is the transformation selected at time step t, $N_t(f)$ is the number of times transformation f has been selected before time step t and c is UCB's exploration coefficient. Next we collect the rollouts



```
Algorithm 2 UCB-DrAC
1: Hyperparameters: Set of image transformations $\mathcal{F} = \{f^1, \ldots, f^n\}$, exploration coefficient c, window for estimating the Q-functions W, number of updates K, initial policy parameters $\pi_\theta$, initial value function $V_\phi$.
2: $N(f) = 1, \forall f \in \mathcal{F}$    ▷ Initialize the number of times each augmentation was selected
3: $Q(f) = 0, \forall f \in \mathcal{F}$    ▷ Initialize the Q-functions for all augmentations
4: $R(f) = \text{FIFO}(W), \forall f \in \mathcal{F}$    ▷ Initialize the lists of returns for all augmentations
5: for $k = 1, \ldots, K$ do
6:     $f_k = \text{argmax}_{f \in \mathcal{F}} \left[ Q(f) + c\sqrt{\frac{\log(k)}{N(f)}} \right]$    ▷ Use UCB to select an augmentation
7:     Update the policy and value function according to Algorithm 1 with $f = f_k$ and $K = 1$:
8:     $\theta \leftarrow \arg\max_\theta J_{\text{DrAC}}$    ▷ Update the policy
9:     $\phi \leftarrow \arg\max_\phi J_{\text{DrAC}}$    ▷ Update the value function
10:    Compute the mean return obtained by the new policy $r_k$.
11:    Add $r_k$ to the $R(f_k)$ list using the first-in-first-out rule.
12:    $Q(f_k) \leftarrow \frac{1}{|R(f_k)|} \sum_{r \in R(f_k)} r$
13:    $N(f_k) \leftarrow N(f_k) + 1$
14: end for
```

Figure 2: UCB-DrAC Algorithm [6]

with the new policy and update the Q value function:

$$Q_t(f) = \frac{1}{K} \sum_{i=t-K}^{t} \mathcal{R}(f_i = f)$$

This is computed as a sliding window average of the past K mean returns obtained by the agent after being updated using augmentation f.

5.2 ProcGen Benchmarks

ProcGen Benchmark [2] consists of 16 unique environments or games designed to measure both sample efficiency and generalization in reinforcement learning. This benchmark is ideal for evaluating generalization since distinct training and test sets can be generated in each environment. The environments' intrinsic diversity demands that agents learn robust policies; overfitting to narrow regions in state space will not suffice. Put differently, the ability to generalize becomes an integral component of success when agents are faced with ever-changing levels. Each of the 16 games corresponds to a distribution of POMDPs q(m), and each level of a game corresponds to a POMDP sampled from that game's distribution m $\sim$ q. The POMDP m is determined by the seed (i.e. integer) used to generate the corresponding level.

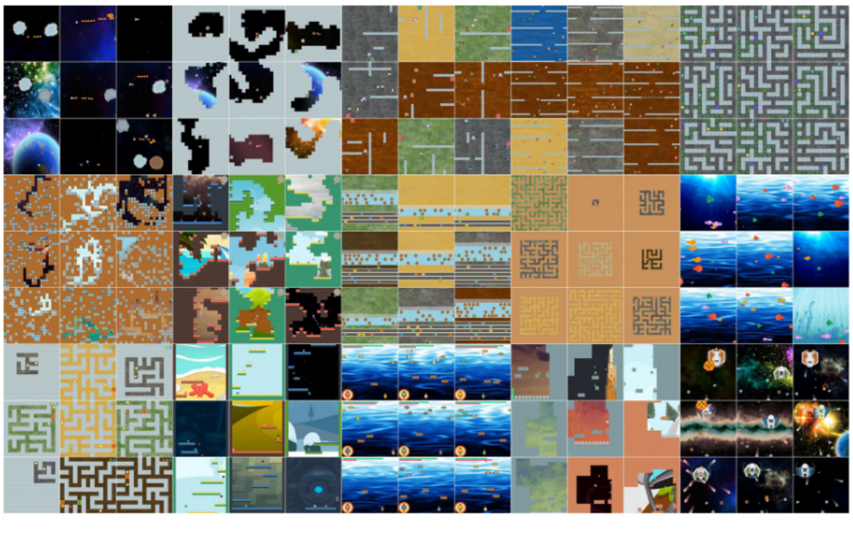


Figure 3: Some examples of the various levels of the ProcGen environments [6]

6 What We Are Going To Do

Given the current framework for automatic data augmentation, we would like to propose the following changes which could make the **generalisation** of the network, better:

6.1 Replacing KL Divergence in G_π with JS Divergence

The Jensen-Shannon (JS) Divergence is given as :

$$JS(P||Q) = 1/2 \times KL(P||M) + 1/2 \times KL(Q||M)$$

where $M = (P + Q)/2$ As discussed in Sec. 4, the UCB-DrAC uses KL Divergence for regularising the policy G_π. The KL Divergence (KL(P || Q)) is known to punish any sample x that obeys :

$$\begin{gathered} Q(x) = 0 \\ P(x) \neq 0 \end{gathered}$$

This is due to the negative log term in the formula for KL Divergence, which makes the entire expression go to infinity. This means that any sample that wasnt encountered in the current distribution, but was present in the target distribution is penalised heavily.
The JS Divergence on the other hand is symmetric and much smoother than the KL Divergence. It is known to be a good loss function for generating samples in the generator part of the GANs as shown by [3] as well. This shows that it has good (maybe even better) generalisation capabilities. In this case, even if any sample present in the target distribution is not encountered in the current distribution, the penalty will be finite ($\frac{q(x)log2}{2}$). Due to this leniency we believe that it would be helpful to improve the generalisation of the existing policy.

6.2 Replacing L_2 Norm in G_v With Elastic Net Based Regularisation

L_2 norm is known to punish the outliers heavily as compared to L_1 norm. On the other hand, the L_1 norm is known to induce sparsity in the corresponding the network, which in turn encourages better generalisations in the model as shown in [5]. However we would also like to penalise any state value that is very far from the un-transformed state values to not affect the subsequent policy significantly. Hence we propose to use the elastic net regularisation [10] instead of the current L_2 regularisation. It is given as :

$$\begin{aligned} Loss = \alpha L_1 + (1 - \alpha) L_2 \\ \alpha \in [0, 1] \end{aligned}$$

Here α is a hyperparameter which will be tuned according to the results.

6.3 Propose Thompson Sampling for Selecting Augmentations

W e would also like to explore more recent methods for selecting an augmentation at each step of training. Currently this is being done by using an epsilon-greedy strategy as described in Sec 5.1 We would like to explore the use of Thompson Sampling which is known to achieve optimal regret (sub-linear) as compared to epsilon greedy which achieves linear regret. This is shown in [1] to succeed empirically as well.

References

[1] Olivier Chapelle and Lihong Li. “An Empirical Evaluation of Thompson Sampling”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper/2011/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf.

[2] Karl Cobbe et al. “Leveraging Procedural Generation to Benchmark Reinforcement Learning”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 2048–2056. URL: https://proceedings.mlr.press/v119/cobbe20a.html.

[3] Ferenc Huszár. “How (not) to train your generative model: Scheduled sampling, likelihood, adversary?” In: *ICLR pre-print* (2016).

[4] Misha Laskin et al. “Reinforcement Learning with Augmented Data”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 19884–19895. URL: https://proceedings.neurips.cc/paper/2020/file/e615c82aba461681ade82da2da38004a-Paper.pdf.

[5] Andrew Ng. “Feature selection, L 1 vs. L 2 regularization, and rotational invariance”. In: *Proceedings of the Twenty-First International Conference on Machine Learning* (Sept. 2004). DOI: 10.1145/1015330.1015435.

[6] Roberta Raileanu et al. “Automatic Data Augmentation for Generalization in Deep Reinforcement Learning”. In: *NeurIPS 2021, Pre-Proceedings* abs/2006.12862 (2020). URL: https://proceedings.neurips.cc/paper/2021/file/2b38c2df6a49b97f706ec9148ce48d86-Paper.pdf.

[7] John Schulman et al. “Proximal Policy Optimization Algorithms.” In: *CoRR* abs/1707.06347 (2017). URL: http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SchulmanWDRK17.

[8] John Schulman et al. “Trust Region Policy Optimization”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1889–1897. URL: https://proceedings.mlr.press/v37/schulman15.html.

[9] 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.

[10] Hui Zou and Trevor Hastie. “Regularization and Variable Selection via the Elastic Net”. In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320. ISSN: 13697412, 14679868. URL: http://www.jstor.org/stable/3647580.