GENERALIZABLE REPRESENTATIONS FOR REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The ability to autonomously learn behaviors via direct interactions in uninstrumented environments can lead to generalist robots capable of enhancing productivity or providing care in unstructured settings like homes. Such uninstrumented settings warrant operations only using the robot's proprioceptive sensor such as onboard cameras, joint encoders, etc which can be challenging for policy learning owing to the high dimensionality and partial observability issues. While these issues can be circumvented by leveraging compressed represenations, such representations acquired via self-supervised techniques are often brittle to task variations and donot generalize across tasks. We propose RRL: Resnet as a representation for Reinforcement Learning – a straightforward yet effective approach that can learn complex behaviors directly from proprioceptive inputs. RRL fuses features extracted from pre-trained Resnet into the standard reinforcement learning pipeline and deliver results comparable to learning directly from the state. In a simulated dexterous manipulation benchmark, where the state of the art methods fails to make significant progress, RRL delivers contact rich behaviors. Its effectiveness in learning behaviors directly from visual inputs with performance and sample efficiency matching learning directly from the state, even in complex high dimensional domains, is far from obvious.

1 INTRODUCTION

Recently, Reinforcement learning (RL) has seen tremendous momentum and progress Mnih et al. (2015); Silver et al. (2017); Jaderberg et al. (2019); Espeholt et al. (2018) in learning complex behaviors from states Schulman et al. (2017); Haarnoja et al. (2018); Rajeswaran et al. (2017). Most success stories, however, are limited to simulations or instrumented laboratory conditions as real world doesn't provide direct access to its internal state. Not only learning with state-space, but visual observation spaces have also found reasonable successKalashnikov et al. (2018); OpenAI et al. (2019). However the sample complexity for methods that can learn directly with visual inputs is far too extreme for direct real-world training Duan et al. (2016); Kaiser et al. (2020). In order to deliver the promise presented by data-driven techniques, we need efficient techniques that can learn unobtrusively learn without environment instrumentation.

Learning without environment instrumentation, especially in unstructured settings like homes, can be quite challenging Zhu et al. (2020); Dulac-Arnold et al. (2019); Ahn et al. (2020). Challenges include – (a) Decision making with incomplete information owing to partial observability as the agents must rely only on proprioceptive on-board sensors (vision, touch, joint position encoders, etc) to perceive and act. (b) The influx of sensory information makes the input space quite high dimensional. (c) Information contamination due to sensory noise and task-irrelevant conditions like lightning, shadows, etc. (d) Most importantly, the scene being flushed with information irrelevant to the task (background, clutter, etc). Agents learning under these constraints is forced to take a large number of samples simply to untangle these task-irrelevant details before it makes any progress on the true task objective. A common approach to handle these high dimensionality and multi-modality issues is to learn representations that distill information into low dimensional features and use them as inputs to the policy (Figure 2). While such ideas have found reasonable success Qin et al. (2019); Manuelli et al. (2019a), designing such representations require a deep understanding of the problem and domain expertise. An alternative approach is to leverage unsupervised representation learning to autonomously acquire representations based on either reconstruction Higgins et al. (2016); Zhu et al.

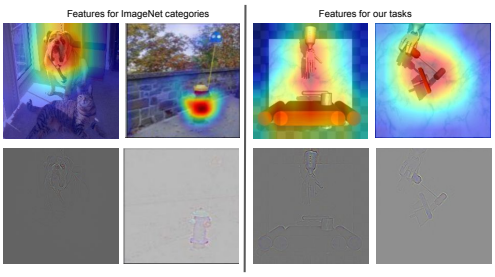

Figure 1: Visualization of Layer 4 of Resnet model of the top 1 class using Grad-CAM Selvaraju et al. (2019)[Top] and Guided Back-propogation Springenberg et al. (2015)[Bottom]. This indicates that Resnet is indeed looking for the right features in our task images (right) in spite of such high distributional shift.

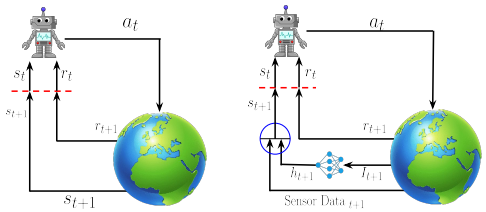

Figure 2: LEFT: An ideal RL setting where the environment outputs the exact state and reward. RIGHT: A practical scenario in which we only have access to the information received from the sensors and the reward. Here, I_{t+1}, Sensor Data$_{t+1}$ represents the information received from the camera and joint motor sensors respectively

(2020); Yarats et al. (2020) or contrastive Srinivas et al. (2020); Stooke et al. (2020) objective. These methods are quite brittle as the representations are acquired from narrow task-specific distributions Stone et al. (2021), and hence, do not generalize well across different tasks. Additionally, they acquire task-specific representations, often needing additional samples from the environment or domains specific data-augmentations for training representations, leading to poor sample efficiency.

The key idea behind our method stems from an intuitive observation over the desiderata of a good representation i.e. (a) it should be low dimensional (b) it should be able to capture silent features encapsulating the diversity and the variability present in our real-world task (c) it should be robust to irrelevant information like noise, lighting, viewpoints, etc (d) it should provide effective representation in the entire distribution that a policy can induce. These requirements are quite harsh needing extreme domain expertise to manually design and an abundance of samples to automatically acquire. Can we acquire this representation without any additional effort? Our work takes a very small step in this direction.

The key insight behind our method is embarrassingly simple – representations do not necessarily have to be trained on the exact task distribution; a representation trained on a *sufficiently* wide distribution of real-world scenarios, will remain effective on any distribution a policy optimizing a task in the real world might induce. While training over such wide distribution is demanding, this is precisely what the success of large image classification models He et al. (2015); Simonyan & Zisserman (2015); Tan & Le (2020); Szegedy et al. (2015) in Computer Vision delivers – representations learned over a large family to real-world scenarios.

Our Contributions: We list the major contributions of our work below-

1. We present a surprisingly simple method (RRL) at the intersection of representation learning, imitation leaning (IL) and reinforcement learning (RL) that uses features from pre-trained image classification models (Resnet34) as representations in standard RL pipeline. Our method is quite general and can be incorporated with minimal changes to state based RL/IL algorithms.

2. Task-specific representations learned by supervised as well as unsupervised methods are usually brittle and suffer from distribution mismatch. We demonstrate that features learned by image classification models are general towards different tasks, robust to visual distractors, and when used in conjunction with standard IL and RL pipelines can efficiently acquire policies directly from proprioceptive inputs.

3. While competing methods have restricted results primarily to planar tasks devoid of depth perspectives, on a rich collection of simulated high dimensional dexterous manipulation tasks, where state-of-the-art methods struggle, we demonstrate that RRL can learn rich behaviors directly from visual inputs with performance & sample efficiency approaching state-based methods.

2 RELATED WORK

RRL rests on recent developments from the fields of Representation Learning, Imitation Learning and Reinforcement Learning. In this section, we outline related works leveraging representation learning for visual reinforcement and imitation learning.

2.1 LEARNING WITHOUT EXPLICIT REPRESENTATION

A common approach is to learn behaviors in an end to end fashion – from pixels to actions – without explicit distinction between feature representation and policy representations. Success stories in this categories range from seminal work Mnih et al. (2013) mastering Atari 2600 computer games using only raw pixels as input, to Levine et al. (2016) which learns trajectory-centric local policies using Guided Policy Search Levine & Koltun (2013) for diverse continuous control manipulation tasks in the real world learned directly from camera inputs. More recently, Haarnoja et al. has demonstrated success in acquiring multi-finger dexterous manipulation Zhu et al. (2018) and agile locomotion behaviors using off-policy action critic methods Haarnoja et al. (2018). While learning directly from pixels has found reasonable success, it requires training large networks with high input dimensionality. Agents require a prohibitively large number of samples to untangle task-relevant information in order to acquire behaviors, limiting their application to simulations or constrained lab settings. RRL maintains an explicit representation network to extract low dimensional features. Decoupling representation learning from policy learning delivery results in large gains in efficiency. Next, we outline related works that use explicit representations.

2.2 LEARNING WITH SUPERVISED REPRESENTATIONS

Another approach is to first acquire representations using expert supervision, and use features extracted from representation as inputs in standard policy learning pipelines. A predominant idea is to learn representative keypoints encapsulating task details from the input images and using the extracted keypoints as a replacement of the state information Kulkarni et al. (2019). Using these techniques, Qin et al. (2019); Manuelli et al. (2019b) demonstrated tool manipulation behaviors in rich scenes flushed with task-irrelevant details. Nagabandi et al. (2019) demonstrated simultaneous manipulation of multiple objects in the task of Baoding ball tasks on a high dimensional dexterous manipulation hand. Along with the inbuilt proprioceptive sensing at each joint, they use an RGB stereo image pair that is fed into a separately pre-trained tracker to produce 3D position estimates You et al. (2020) for the two Baoding balls. These methods, while powerful, learn task-specific features and requires expert supervision, making it harder to (a) translate to variation in tasks/environments, (b) scale with increasing task diversity. RRL, on the other hand, uses single task-agnostic representations making it easy to scale.

2.3 LEARNING WITH UNSUPERVISED REPRESENTATIONS

With the ambition of being scalable, this group of methods intends to acquire representation via unsupervised techniques. Sermanet et al. (2018) uses contrastive learning to time-align visual features across different embodiment demonstration behavior transfer from human to a Fetch robot. Burgess et al. (2018), Finn et al.; Zhu et al. (2020) use variational inference Kingma & Welling (2014); Burgess et al. (2018) to learn compressed latent representations and use it as input to standard RL pipeline to demonstrate rich manipulation behaviors. Hafner et al. (2020) additionally learns dynamics models directly in the latent space and used model-based RL to acquire behaviors on simulated tasks. On similar tasks, Hafner et al. (2019) uses multi-step variational inference to learn world dynamic as well as rewards models for off-policy RL. Srinivas et al. (2020) uses image augmentation with variations inference to construct features to be used in standard RL pipeline and demonstrated performance at par with learning directly from the state. Laskin et al. (2020); Kostrikov et al. (2020) demonstrate comparable results by assimilating updates over features acquired only via image augmentation. Similar to supervised methods, unsupervised methods often learns task-specific brittle representations and often suffers challenges from non-stationarity arising from the mismatch between the distribution representations are learned on and the distribution policy induces. To induce stability, RRL uses pre-trained stationary representations trained on distribution with wider support than what policy can induce. Additionally, representations learned over a wide distribution

of real-world samples are robust to noise and irrelevant information like lighting, illumination, etc. Unlike prior works Hafner et al. (2020; 2019); Srinivas et al. (2020); Laskin et al. (2020); Kostrikov et al. (2020), which demonstrate results on simple planner task devoid of depth perspective, we demonstrate RRL on a suite of high dimensional contact rich dexterous manipulation tasks.

2.4 Learning with representations and demonstrations

Learning from demonstrations has a rich history. We focus our discussion on DAPG Rajeswaran et al. (2017), a state-based Natural policy gradient Rajeswaran et al. (2018) method which optimized for the natural gradient of a joint loss with imitation as well as reinforcement objective. DAPG has been demonstrated to outperform competing methods Gupta et al. (2017); Hester et al. (2017) on the high dimensional ADROIT dexterous manipulation task suite we test on. RRL extends DAPG to solve the task suite directly from proprioceptive signals with performance and sample efficiency comparable to state-DAPG. Unlike DAPG which is on-policy, FERM Zhan et al. (2020) is a closely related off-policy actor-critic methods combining learning from demonstrations with RL. FERM builds on RAD Laskin et al. (2020) and inherits its challenges. We demonstrate via experiments that RRL is more stable, more robust to various distractors, and convincingly outperforms FERM.

3 Background

RRL solves a standard Markov decision process (Section 3.1) by combining three fundamental building blocks. (a) Policy gradient algorithm (Section 3.2), (b) Demonstration bootstrapping (Section 3.3), and (c) Representation learning (Section 3.4). We briefly outline these fundamentals before detailing with our method in Section 4.

3.1 Preliminaries: MDP

We model the control problem as a Markov decision process (MDP), which is defined using the tuple: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \rho_0, \gamma)$. $\mathcal{S} \in \mathbb{R}^n$ and $\mathcal{A} \in \mathbb{R}^m$ represent the state and actions. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. In the ideal case, this function is simply an indicator for task completion (*sparse reward setting*). $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition dynamics, which can be stochastic. In model-free RL, we do not assume any knowledge about the transition function and require only sampling access to this function. ρ_0 is the probability distribution over initial states and $\gamma \in [0, 1)$ is the discount factor. We wish to solve for a stochastic policy of the form $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which optimizes the expected sum of rewards:

$$\eta(\pi) = \mathbb{E}_{\pi,\mathcal{M}}\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big] \tag{1}$$

3.2 Policy Gradient

The goal of the RL agent is to maximise the expected discounted return $\eta(\pi)$ [Equation 1] under the distribution induced by the current policy π. Policy Gradient algorithms optimize the policy ($\pi_\theta(a \mid s)$) directly, where θ is the function parameter by estimating $\nabla\eta(\pi)$. First we introduce a few standard notations, Value function : $V^\pi(s)$, Q function : $Q^\pi(s, a)$ and the advantage function : $A^\pi(s, a)$. The advantage function can be considered as another version of Q-value with lower variance by taking the state-value off as the baseline.

$$V^\pi(s) = \mathbb{E}_{\pi\mathcal{M}}\Big[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\Big] \tag{2}$$

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}_{\mathcal{M}}\Big[\mathcal{R}(s, a)\Big] + \mathbb{E}_{s' \sim \mathcal{T}(\int, \dashv)}\Big[V^\pi(s')\Big] \\ A^\pi(s, a) &= Q^\pi(s, a) - V^\pi(s) \end{aligned} \tag{3}$$

The gradient can be estimated using the Likelihood ratio approach and Markov property of the problem Williams (1992) and using a sampling based strategy,

$$\nabla\eta(\pi) = \; g = \frac{1}{NT}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \; log \; \pi_\theta(a_t^i|s_t^i)\hat{A}^\pi(s_t^i, a_t^i, t) \tag{4}$$

Amongst the wide collection of policy gradient algorithms, we build upon Natural Policy Gradient (NPG) Rajeswaran et al. (2018) to solve our MDP formulation owing to its stability and effectiveness in solving complex problems, especially in high dimensional domains Rajeswaran et al. (2018). We refer to Weng for a detailed background on different policy gradient approaches. Next, we describe how human demonstrations can be effectively used along with NPG to aid policy optimization.

3.3 Demo Augmented Policy Gradient

Policy Gradients with appropriately shaped rewards can solve arbitrarily complex tasks. However, real-world environments seldom provide a shaped reward, and must be manually specified by domain experts. Learning with sparse signals, such as task completion indicator functions can relax domain expertise in reward shaping but it results in extremely high sample complexity due to exploration challenges. DAPG (Rajeswaran et al.) combines policy gradients with few demonstrations to curb these challenges.

We represent the demonstration dataset using $\rho_D = \left\{\left(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)}, r_t^{(i)}\right)\right\}$ where t indexes time and i indexes different trajectories. DAPG incorporates demonstrations in two way to efficiently learn from the them :

(1) Warm up the policy using few demonstrations (25 in our setting) using a simple Mean Squared Error(MSE) loss, i.e, initialize the policy using behavior cloning [5]. This provides an informed policy initialization that helps in resolving the exploration issue and reduces the sample complexity.

$$L_{BC}(\theta) = \frac{1}{2}\sum_{i,j\in minibatch}\left(\pi_\theta(s_t^{(i)}) - a_t^{(i)H}\right)^2 \tag{5}$$

where, θ are the agent parameters and $a_t^{(i)H}$ represents the action taken by the human.

(2) DAPG builds upon on-policy NPG algorithm Rajeswaran et al. (2018) which uses a normalized gradient ascent procedure where the normalization is under the Fischer metric.

$$\theta_{k+1} = \theta_k + \sqrt{\frac{\delta}{g^T\hat{F}_{\theta_k}^{-1}g}}\hat{F}_{\theta_k}^{-1}g \tag{6}$$

where $\hat{F}_{\theta_k}$ is the Fischer Information Metric at the current iterate θ_k,

$$\hat{F}_{\theta_k} = \frac{1}{T}\sum_{t=0}^{T}\nabla_\theta log \;\; \pi_\theta(a_t|s_t)\nabla_\theta log \; \pi_\theta(a_t|s_t)^T \tag{7}$$

and g is the sample based estimate of the policy gradient [4]. To make the best use of available demonstrations, DAPG proposes a joint loss g_{aug} with task as well as imitation objective. The imitation objective asymptotically decays over time allowing the agent to learn behaviors surpassing the expert.

$$\begin{aligned} g_{aug} = & \sum_{(s,a)\in\rho_\pi}\nabla_\theta \; ln \; \pi_\theta(a|s)A^\pi(s,a) \\ & + \sum_{(s,a)\in\rho_D}\nabla_\theta \; ln \; \pi_\theta(a|s)w(s,a) \end{aligned} \tag{8}$$

where, ρ_π is the dataset obtained by executing the current policy, ρ_D is the demonstration data and $w(s,a)$ is the heuristic weighting function defined as :

$$w(s,a) = \lambda_0\lambda_1^k \max_{(s',a')\in\rho_\pi} A^\pi(s',a') \quad \forall \;\; (s,a)\in\rho_D \tag{9}$$

Though this work builds upon Natural Policy Gradients, in principle, any state based RL algorithm, can be integrated with our approach. In the next section, we outline our method that scales DAPG towards visual information.

3.4 REPRESENTATION LEARNING

Until now in the above sections, we have discussed assuming direct access to the low-level state information. In a real-world scenario, access to such state information is infeasible, instead, we have access to high dimensional observations like images. DAPG has thus far only been demonstrated to be effective with low-level state. DAPG is based on NPG which works well but faces issues with input dimensionality and hence, cannot be directly used with the images. Representation learning Bengio et al. (2014) is learning representations of input data typically by transforming it or extracting features from it, which makes it easier to perform the task (in our case it can be used in place of the exact state of the environment). These representation learning techniques can be used to tackle the curse of dimensionality in an intuitive way Wulfmeier et al. (2020). The feature extractor can be (a) Learned along with RL objective Srinivas et al. (2020); Yarats et al. (2020). (b) Decoupled from RL by pre-training itStooke et al. (2020). The former lead to task-specific features that can get quite narrow and specific to the domain. This can hurt the generalization performance especially when it has to perform in diverse environmental settings. Thus, we pre-train the feature extractor initially and use it across all the tasks that we have considered. Next we describe RRL in detail.

4 RRL: RESNET AS REPRESENTATION FOR RL

In an ideal RL setting, the agent interacts with the environment based on the current state, and in return, the environment outputs the next state and the reward obtained. Refer to the left side of Figure 2. This works well in a simulated environment but in a real-world scenario, we do not have access to this low-level state information. Instead we get the information from cameras (I_t) and other sensors like encoders (Sensor Data $_t$). To overcome the challenges associated with learning with high dimensional spaces (highlighted in section 3.4), we use representations that project information into a lower-dimensional manifolds.

RRL uses the standard Resnet-34 model as our feature extractor. Resnet is trained over a wide variety of real-world classes. We demonstrate that the diversity of the Resnet feature extractor allows us to use it across all tasks we considered. This further simplifies our algorithm as our representations are frozen and do-not face the non-stationarity issues encountered while learning policy and representations in tandem.

The features (h_t) obtained from the above feature extractor are appended with the information obtained from the internal joint encoders of the Adroit Hand (Sensor Data $_t$). We show that this can be used as a substitute for the exact state(s_t) of the agent in the RL algorithm.

$$s_t = [h_t, \text{Sensor Data }_t] \tag{10}$$

here $h_t = f(I_t) \in \mathbb{R}^k$, $I_t \in \mathbb{R}^n$ represents the raw image obtained from the camera where $k << n$, and f is the feature extractor (Resnet). [Pseudo-code - 1].

5 EXPERIMENTAL EVALUATIONS

Our experimental evaluations aims to address the following questions: (1) Can RRL learn complex tasks directly from proprioceptive signals (camera inputs and joint encoders) , (2) How various representational choices influence the generality and versatility of the resulting behaviors, (3) How does RRL's performance and efficiency compare against other state-of-the-art methods, (4) What are the effects of various design decisions on our method?

5.1 TASKS

While the applicability of prior proprioception based RL methods Laskin et al. (2020); Kostrikov et al. (2020); Hafner et al. (2020) have been limited to simple low dimensional tasks (Cartpole swing-up, Cheetah run, Reacher, Finger spin, Walker walk, Ball in cup catch, etc), we demonstrate performance on ADROIT manipulation suite Rajeswaran et al. (2017) which consists of high dimensional dexterous manipulation tasks (Figure7). Furthermore, unlike prior task sets, which are fundamentally planner tasks devoid of depth perspective, the ADROIT manipulation suite consists of visually rich physically realistic tasks demanding representations untangling depth information as well.

5.2 RESULTS

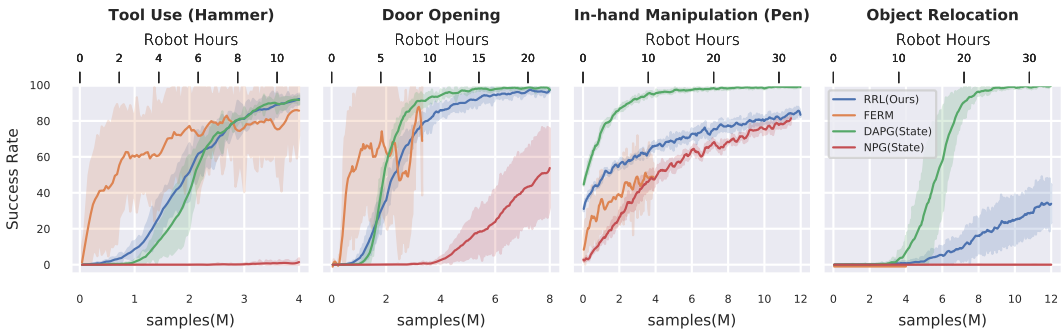

Figure 3: Performance on ADROIT dexterous manipulation suite Rajeswaran et al. (2017): State of the art policy gradient method NPG_{state} Rajeswaran et al. (2018) struggles to solve the suite even with privileged low level state information, establishing the difficulty of the suite. Amongst demonstration accelerated methods, RRL(ours) demonstrates stable performance and approaches performance of $DAPG_{state}$ Rajeswaran et al. (2017) (upper bound), a demonstration accelerated method using privileged state information. A competing baseline FERM Zhan et al. (2020) makes good initial, but unstable, progress in a few tasks and often saturates in performance before exhausting our computational budget (40 hours/ task/ seed).

In Figure 3, we contrast the performance of RRL against the state of the art baselines. We observed that NPG Kakade (2001) struggles to solve the suite even with full state information, which establishes the difficulty of our task suite. DAPG$_{state}$ Rajeswaran et al. (2017) uses privileged state information from the environment to solve the tasks and pose as the best case oracle. RRL demonstrates good performance on most task, relocate being the hardest, and often approaches performance comparable to DAPG$_{state}$.

A competing baseline FERM [1] Zhan et al. (2020) is quite unstable in these tasks. It starts strong for hammer and door tasks but saturates in performance. It makes slow progress in pen, and completely fails for relocate.

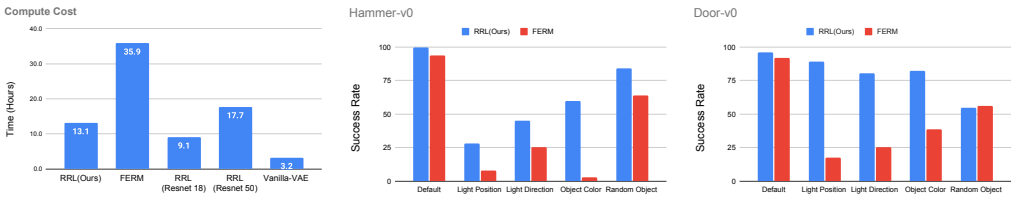

Figure 4: LEFT: Comparison of the computational cost of RRL(Ours) - Resnet 34, FERM - Strongest baseline, RRL with Resnet 18, RRL with Resnet 50 and RRL(VAE) baseline. CENTER,RIGHT: Influence of various environment distractions, lightning condition, object color on RRL(Ours), and FERM. RRL(Ours) consistently performs better than FERM in all the variations

In Figure 4 we compare the computational footprint of various methods we consider. We note that our method is approximately three times more compute-efficient than FERM (and other variations we consider). In Figure 4 we probe the robustness of the final policies by injecting various distractors in the environment during inference. We note that the resilience of the resnet features induces robustness to RRL's policies. Task-specific features learned by FERM are brittle leading to larger degradation in performance.

5.3 EFFECTS OF REPRESENTATION

In figure 5, we study the effects various representational choices. We note that RRL using resnet features significantly outperforms a variant *RRL(VAE)* (see appendix for details) using features learned via variational inference Yarats et al. (2020); Ha & Schmidhuber (2018a;b); Higgins et al.

[1] Reporting best performance amongst over 30 configurations per task we tried in consultation with the FERM authors.

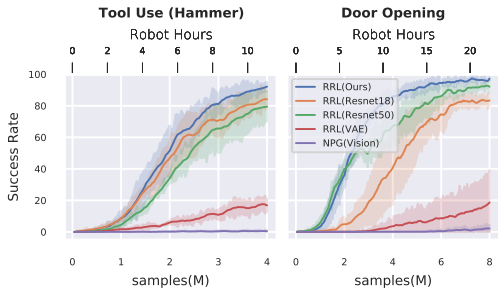

Figure 5: Influence of representation: RRL(Ours), using resnet34 features, outperforms commonly used representation learning method VAE Yarats et al. (2020); Ha & Schmidhuber (2018a;b); Higgins et al. (2018); Nair et al. (2018). Amongst different Resnet variations, Resnet34 strikes the balance between representation capacity and computational overhead. NPG$_{vision}$ showcases the performance with Resnet features but without demonstration bootstrapping, indicating that only representational choices are not enough to solve the task suite.

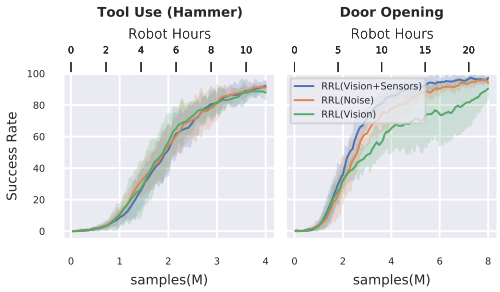

Figure 6: Influence of proprioceptive signals on RRL(Ours): RRL$_{vision}$ demonstrates that RRL remains performant with (only) visual inputs as well. RRL$_{noise}$ demonstrates that RRL remains effectiveness inpresence of noisy (2%) proprioception

(2018); Nair et al. (2018) techniques. This indicates that pre-trained Resnet provides superior features compared to methods that explicitly learn task-specific features using additional samples from the environment. Unlike features learned from the narrow distribution of task environments, Resnet features are trained on diverse real-world categories. In addition to improved sample complexity resulting from the use of pre-trained features, the resilience, robustness, and versatility of Resnet features lead to policies that are also robust to visual distractors and clutter in the scene. In Figure 4 we observe that RRL is quite resilient to visual distractions (light, colors, redundant objects) encountered during test cases.

5.4 Effects of proprioception choices and sensor noise

While it's hard to envision a robot without proprioceptive joint sensing, harsh conditions of the real-world can lead to noisy sensing, even sensor failures. In Figure 6, we subjected RRL to (a) signals with 2% noise in the information received from the joint encoders, and (b) only visual inputs are used as proprioceptive signals. In both these cases, our methods remained performant with slight to no degradation in performance.

6 Discussion: Limitations & Opportunities

This paper presents an intuitive idea bringing together advancements from the fields of representation learning, imitation learning, and reinforcement learning. A very simple method named RRL that leverages Resnet features as representation to learn complex behaviors directly from proprioceptive signals. The resulting algorithm approaches the performance of state-based methods in complex ADROIT dexterous manipulation suite.

Limitations: While this work demonstrates promises of using pre-trained features, it doesn't investigate the data mismatch problem that might exist. Real-world datasets used to train resnet features are from human-centric environments. While we desire robots to operate in similar settings, there are still differences in their morphology and mode of operations. Additionally, resent (and similar models) acquire features from data primarily comprised of static scenes. In contrast, embodied agents desire rich features of dynamic and interactive movements.

Opportunities: RRL uses a single pre-trained representation for solving four complex and very different tasks. Unlike the domains of vision and language, there is a non-trivial cost associated with data in robotics. The possibility of having a standard shared representational space opens up avenues for (a) leveraging data from various sources, (b) building hardware-accelerated devices from feature compression (c) low latency information transmission.

REFERENCES

Ahn, M., Zhu, H., Hartikainen, K., Ponte, H., Gupta, A., Levine, S., and Kumar, V. Robel: Robotics benchmarks for learning with low-cost robots. In *Conference on Robot Learning*, pp. 1300–1313. PMLR, 2020.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives, 2014.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. Understanding disentangling in β-vae, 2018.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning, 2016.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders.

Gupta, A., Eppner, C., Levine, S., and Abbeel, P. Learning dexterous manipulation for a soft robotic hand from human demonstration, 2017.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution, 2018a.

Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2018b.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications, 2019.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels, 2019.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J. Z., and Gruslys, A. Deep q-learning from demonstrations, 2017.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Higgins, I., Pal, A., Rusu, A. A., Matthey, L., Burgess, C. P., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. Darla: Improving zero-shot transfer in reinforcement learning, 2018.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castañeda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., and et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, May 2019. ISSN 1095-9203. doi: 10.1126/science.aau6249. URL `http://dx.doi.org/10.1126/science.aau6249`.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., and Michalewski, H. Model-based reinforcement learning for atari, 2020.

Kakade, S. M. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2014.

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels, 2020.

Kulkarni, T., Gupta, A., Ionescu, C., Borgeaud, S., Reynolds, M., Zisserman, A., and Mnih, V. Unsupervised learning of object keypoints for perception and control, 2019.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data, 2020.

Levine, S. and Koltun, V. Guided policy search. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1–9, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL http://proceedings.mlr.press/v28/levine13.html.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies, 2016.

Manuelli, L., Gao, W., Florence, P., and Tedrake, R. kpam: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2019a.

Manuelli, L., Gao, W., Florence, P., and Tedrake, R. kpam: Keypoint affordances for category-level robotic manipulation, 2019b.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Nagabandi, A., Konoglie, K., Levine, S., and Kumar, V. Deep dynamics models for learning dexterous manipulation, 2019.

Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals, 2018.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. Solving rubik's cube with a robot hand, 2019.

Qin, Z., Fang, K., Zhu, Y., Fei-Fei, L., and Savarese, S. Keto: Learning keypoint representations for tool manipulation, 2019.

Rajeswaran, A., Kumar, V., Gupta, A., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *CoRR*, abs/1709.10087, 2017. URL http://arxiv.org/abs/1709.10087.

Rajeswaran, A., Lowrey, K., Todorov, E., and Kakade, S. Towards generalization and simplicity in continuous control, 2018.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. Trust region policy optimization, 2017.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL http://dx.doi.org/10.1007/s11263-019-01228-7.

Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., and Levine, S. Time-contrastive networks: Self-supervised learning from video, 2018.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017. URL http://dx.doi.org/10.1038/nature24270.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition, 2015.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net, 2015.

Srinivas, A., Laskin, M., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning, 2020.

Stone, A., Ramirez, O., Konolige, K., and Jonschkowski, R. The distracting control suite – a challenging benchmark for reinforcement learning from pixels, 2021.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning, 2020.

Subramanian, A. Pytorch-vae. https://github.com/AntixK/PyTorch-VAE, 2020.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision, 2015.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

Weng, L. Policy gradient algorithms. *lilianweng.github.io/lil-log*, 2018. URL https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pp. 229–256, 1992.

Wulfmeier, M., Byravan, A., Hertweck, T., Higgins, I., Gupta, A., Kulkarni, T., Reynolds, M., Teplyashin, D., Hafner, R., Lampe, T., and Riedmiller, M. Representation matters: Improving perception and exploration for robotics, 2020.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images, 2020.

You, Y., Lou, Y., Li, C., Cheng, Z., Li, L., Ma, L., Wang, W., and Lu, C. Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations, 2020.

Zhan, A., Zhao, P., Pinto, L., Abbeel, P., and Laskin, M. A framework for efficient robotic manipulation, 2020.

Zhu, H., Gupta, A., Rajeswaran, A., Levine, S., and Kumar, V. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost, 2018.

Zhu, H., Yu, J., Gupta, A., Shah, D., Hartikainen, K., Singh, A., Kumar, V., and Levine, S. The ingredients of real-world robotic reinforcement learning, 2020.

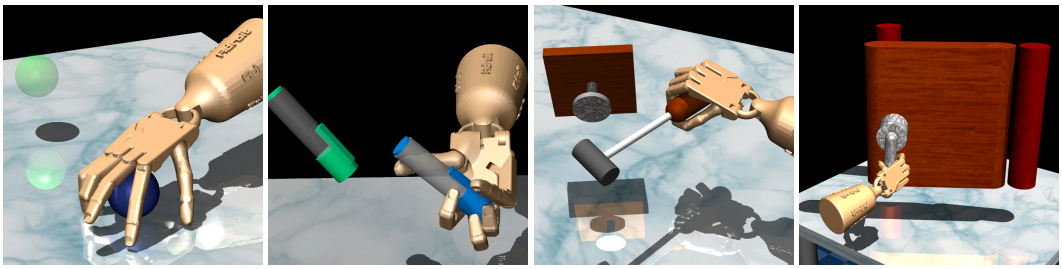

Figure 7: ADROIT manipulation suite consisting of complex dexterous manipulation tasks involving object relocation, in hand manipulation (pen repositioning), tool use (hammering a nail), and interacting with human centric environments (opening a door).

7 APPENDIX

7.1 RRL(OURS)

Same parameters are used across all the tasks (pen-v0, door-v0, hammer-v0 and relocate-v0) unless explicity mentioned. Sparse reward setting is used in all the environments as proposed by Rajeswaran et al.. We have directly used the parameters provided by DAPG without any additional fine tuning except for the policy size (same across all tasks).

Parameters	**Setting**
Encoder	Resnet34
BC batch size	32
BC epochs	5
BC learning rate	0.001
Policy Size	(256, 256)
Samples collected per iteration	40000
vf _batch_size	64
vf_epochs	2
rl_step_size	0.05
rl_gamma	0.995
rl_gae	0.97
lam_0	0.01
lam_1	0.95

7.2 ABLATIONS AND ANALYSIS OF DESIGN DECISIONS

In our next set of experiments, we evaluate the effect of various design decisions on our method. In Figure 5, we study the effect of different Resnet features as our representation. Resnet34, though computationally more demanding (Figure 4) than Resnet18, delivers better performance owing to its improved representational capacity and feature expressivity. A further boost in capacity (Resnet50) degrades performance, likely due to the incorporation of less useful features and an increase in samples required to train the resulting larger policy network.

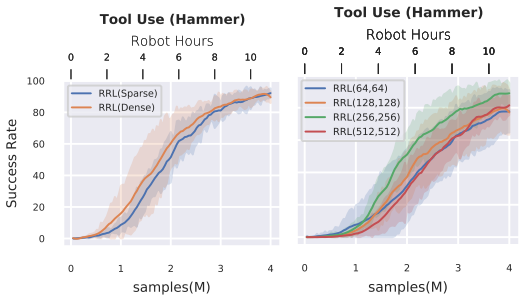

Figure 8: LEFT: Influence of rewards signals: RRL(Ours), using sparse rewards, remains performant with a variation RRL_{dense} using well-shaped dense rewards. RIGHT: Effect of policy size on the performance of RRL . We observe that it is quite stable with respect to a wide range of policy sizes.

Reward design, especially for complex high dimensional tasks, requires significant domain expertise. RRL replaces the needs of well-shaped rewards by using a few demonstrations (to curb the exploration challenges in high dimensional space) and sparse rewards (indicating task completion). This significantly lowers the domain exper-

Algorithm 1 RRL

Input: 25 Human Demonstrations ρ_D
Initialize using Behavior Cloning [Eq.5].
repeat
  for i = 1 to n **do**
    for t = 1 to horizon **do**
      Take action
      $a_t = \pi_\theta([Encoder(I_t), SensorData\ _t])$
      and receive $I_{t+1}, SensorData\ _{t+1}, r_{t+1}$
      from the environment.
    end for
  end for
  Compute $\nabla_\theta\ log\ \pi_\theta(a_t|s_t)$ for each $(s, a) \in \rho_\pi, \rho_D$
  Compute $A^\pi(s, a)$ for each $(s, a) \in \rho_\pi$ and $w(s, a)$ for each $(s, a) \in \rho_D$ according to Equations 2, 9
  Calculate policy gradient according to 8
  Compute Fisher matrix 7
  Take the gradient ascent step according to 6.
  Update the parameters of the value function in order to approximate(2) : $V_k^\pi(s_t^{(n)}) \approx \sum_{t'=t}^{T} \gamma^{t'-t} r_t^{(n)}$
until Satisfactory performance

tise required for our methods. In Figure 8, we observe that RRL (using sparse rewards) delivers competitive performance to a variant of our methods that uses well-shaped dense rewards. In figure, 8 we note that RRL is resilient to variation in policy network capacity.

7.3 Resnet Architectural Details

: We use standard Resnet-34 model as RRL's feature extractor. The model is pre-trained on the ImageNet dataset which consists of 1000 classes. It is trained on 1.28 million images on the classification task of ImageNet. The last layer (fc layer) of the model is removed and all the parameters are frozen throughout the training of the RL agent. During inference, the observations obtained from the environment are of size 256×256, a center crop of size 224×224 is fed into the model. We also evaluate our model using different Resnet sizes (Figure 5).

7.4 RRL(VAE)

Figure 9: ROW1: Original input images of the Hammer task; ROW2: Corresponding Reconstructed images; ROW3: Original input images of the Door task; ROW4: Corresponding Reconstructed images. These images depict that the latent features sufficiently encodes features required to reconstruct the images.

For training, we collected a dataset of 1 million images of size 64 x 64. Out of the 1 million images collected, 25% of the images are collected using an optimal course of actions (expert policy), 25% with a little noise (expert policy + small noise), 25% with even higher level of noise (expert policy + large noise) and remaining portion by randomly sampling actions (random actions). This is to ensure that the images collected sufficiently represents the distribution faced by policy during the training of the agent. We observed that this significantly helps compared to collecting data only from the expert policy. The variational auto-encoder(VAE) is trained using a reconstruction objective Kingma

& Welling (2014) for 10epochs. Figure 9 showcases the reconstructed images. We used a latent size of 512 for a fair comparison with Resnet. The weights of the encoder are freezed and used as feature extractors in place of Resnet in RRL. RRL(VAE) also uses the inputs from the pro-prioceptive sensors along with the encoded features. VAE implementation courtesy Subramanian (2020).

7.5 Visual Distractor Evaluation details

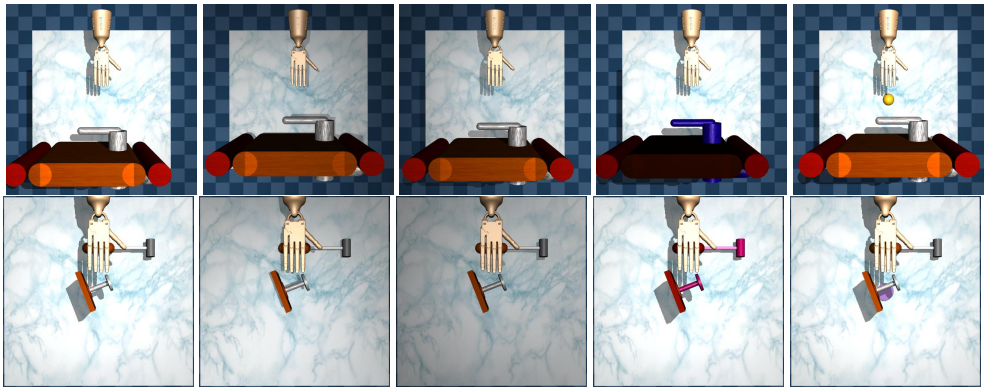

Figure 10: COL1: Original images; COL2: Change in light position; COL3: Change in light direction; COL4: Randomizing object colors; COL5: Introducing a random object in the scene. All the parameters are randomly sampled every time in an episode.

In order to test the generalisation performance of RRL and FERM Zhan et al. (2020), we subject the environment to various kinds of visual distractions during inference (Figure 10). Note all parameters are freezed during this evaluation, an average performance over 75 rollouts is reported. Following distractors were used during inference to test robustness of the final policy -

- Random change in light position.
- Random change in light direction.
- Random object color. (Handle, door color for Door-v0; Different hammer parts and nail for Hammer-v0)
- Introducing a new object in scene - random color, position, size and geometry (Sphere, Capsule, Ellipsoid, Cylinder, Box).

7.6 Compute Cost calculation

We calculate the actual compute cost involved for all the methods (RRL(Ours), FERM, RRL(Resnet-50), RRL(Resnet-18)) that we have considered. Since in a real-world scenario there is no simulation of the environment we do not include the cost of simulation into the calculation. For fair comparison we show the compute cost with same sample complexity (4 million steps) for all the methods. FERM is quite compute intensive (almost 3x RRL(Ours)) because (a) Data augmentation is applied at every step (b) The parameters of Actor and Critic are updated once/twice at every step (Compute results shown are with one update per step) whereas most of the computation of RRL goes in the encoding of features using Resnet. The cost of VAE pretraining in included in the over all cost. RRL(Ours) that uses Resnet-34 strikes a balance between the computational cost and performance. **Note:** No parallel processing is used while calculating the cost.