RRL: Resnet as representation for Reinforcement Learning

Rutav Shah [* 1] **Vikash Kumar** [* 2]

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. We propose RRL: Resnet as 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 delivers 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. The appeal of RRL lies in its simplicity in bringing together progress from the fields of Representation Learning, Imitation Learning, and Reinforcement Learning. 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.,
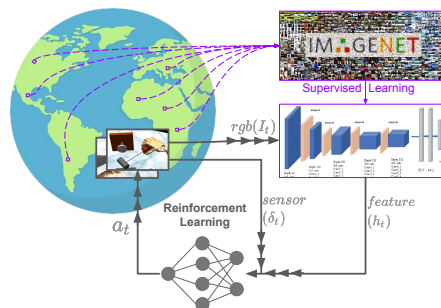

Figure 1. RRL Resnet as representation for Reinforcement Learning takes a small step in bridging the gap between Representation learning and Reinforcement learning. RRL pre-trains an encoder on a wide variety of real world classes like ImageNet dataset using a simple supervised classification objective. Since the encoder is exposed to a much wider distribution of images while pretraining, it remains effective whatever distribution the policy might induce during the training of the agent. This allows us to freeze the encoder after pretraining without any additional efforts.

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 success (Kalashnikov 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 without the need of 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 untan-

---

[*]Equal contribution [1]Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India [2]Department of Computer Science, University of Washington, Seattle, USA. Correspondence to: Rutav Shah <rutavms@gmail.com>, Vikash Kumar <vikash@cs.washington.edu>.

gle 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 distil information into low dimensional features and use them as inputs to the policy. While such ideas have found reasonable success (Qin et al., 2019; Manuelli et al., 2019a), designing such representations in a supervised manner requires 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., 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 Table 1. Additionally, they acquire task-specific representations, often needing additional samples from the environment leading to poor sample efficiency or domains specific data-augmentations for training representations.

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 for a compact representation. (b) it should be able to capture silent features encapsulating the diversity and the variability present in a real-world task for better generalization performance. (c) it should be robust to irrelevant information like noise, lighting, viewpoints, etc so that it is resilient to the changes in surroundings. (d) it should provide effective representation in the entire distribution that a policy can induce for effective learning. 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 (Figure 1) 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 of real-world scenarios.

Our Contributions: We list the major contributions

1. We present a surprisingly simple method (RRL) at the intersection of representation learning, imitation leaning (IL) and reinforcement learning (RL) that uses
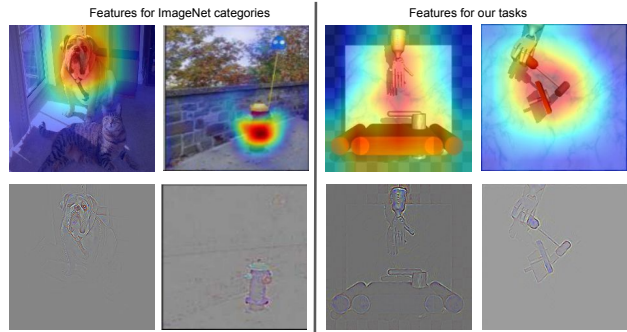

Figure 2. Visualization of Layer 4 of Resnet model of the top 1 class using Grad-CAM (Selvaraju et al., 2019)[Top] and Guided Backpropogation (Springenberg et al., 2015)[Bottom]. This indicates that Resnet is indeed looking for the right features in our task images (right) in spite of such high distributional shift.

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 most 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 task (Figure 2), 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.

4. Additionally, we underline the performance gap between the SOTA approaches and RRL on simple low dimensional tasks as well as high dimensional more realistic tasks. Furthermore, we experimentally establish that the commonly used environments for studying image based continuous control methods are not a true representative of real-world scenario.

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 delivers results with 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 separate 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, and (b) scale with increasing task diversity. RRL, on the other hand, uses single task-agnostic representations with better generalization capability 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 to demonstrate 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 use 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) use image augmentation with variational inference to construct features to be used in standard RL pipeline and demonstrate 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 as they break when subjected to small variations in the surroundings 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.

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 method which optimizes for the natural gradient (Kakade, 2001a) 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 like learning task-specific representations. We demonstrate via experiments that RRL is more stable, more robust to various distractors, and convincingly outperforms FERM since RRL uses a fixed feature extractor pre-trained over wide variety of real world images and avoids learning task specific representations.

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 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.

$$V^\pi(s) = \mathbb{E}_{\pi \mathcal{M}} \Big[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \Big]$$
$$Q^\pi(s, a) = \mathbb{E}_{\mathcal{M}} \Big[\mathcal{R}(s, a) \Big] + \mathbb{E}_{s' \sim \mathcal{T}(\int, \dashv)} \Big[V^\pi(s') \Big]$$
$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$
$$\tag{2}$$

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

$$\nabla \eta(\pi) = g = \frac{1}{NT} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \hat{A}^\pi(s_t^i, a_t^i, t) \tag{3}$$

Amongst the wide collection of policy gradient algorithms, we build upon Natural Policy Gradient (NPG) (Kakade, 2001a) to solve our MDP formulation owing to its stability and effectiveness in solving complex problems. We refer to (Weng, 2018) for a detailed background on different policy gradient approaches. In the next section, 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 shaped rewards, and it 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 in two ways to mitigate this issue and effectively learn from them. 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.

(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 [Eq 4]. This provides an informed policy initialization that helps in resolving the early exploration issue as it now pays attention to task relevant state-action pairs and thereby, reduces the sample complexity.

$$L_{BC}(\theta) = \frac{1}{2} \sum_{i,j \in minibatch} \left(\pi_\theta(s_t^{(i)}) - a_t^{(i)H} \right)^2 \tag{4}$$

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

(2) DAPG builds upon on-policy NPG algorithm (Kakade, 2001a) which uses a normalized gradient ascent procedure where the normalization is under the Fischer metric.

$$\theta_{k+1} = \theta_k + \sqrt{\frac{\delta}{g^T \hat{F}_{\theta_k}^{-1} g}} \hat{F}_{\theta_k}^{-1} g \tag{5}$$

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

$$\hat{F}_{\theta_k} = \frac{1}{T} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) \nabla_\theta \log \pi_\theta(a_t | s_t)^T \tag{6}$$

and g is the sample based estimate of the policy gradient [Eq 3]. To make the best use of available demonstrations,

DAPG proposes a joint loss g_{aug} combining task as well as imitation objective. The imitation objective asymptotically decays over time allowing the agent to learn behaviors surpassing the expert.

$$g_{aug} = \sum_{(s,a) \in \rho_\pi} \nabla_\theta \ln \pi_\theta(a|s) A^\pi(s,a)$$
$$+ \sum_{(s,a) \in \rho_D} \nabla_\theta \ln \pi_\theta(a|s) w(s,a) \tag{7}$$

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 :

$$w(s,a) = \lambda_0 \lambda_1^k \max_{(s',a') \in \rho_\pi} A^\pi(s',a') \quad \forall \ (s,a) \in \rho_D \tag{8}$$

DAPG has proven to be successful in learning policy for the dexterous manipulation tasks with reasonable sample complexity.

3.4. Representation Learning

DAPG has thus far only been demonstrated to be effective with access to low-level state information which is not readily available in real-world. DAPG is based on NPG which works well but faces issues with input dimensionality and hence, cannot be directly used with the input images acquired from onboard cameras. 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). Let $I \in \mathbb{R}^n$ represents the high dimensional input image, then

$$h = f_\rho(I) \tag{9}$$

where f represents the feature extractor, ρ is the distribution over which f is valid and $h \in \mathbb{R}^d$ with $d << n$ is the compact, low dimensional representation of I. In the next section, we outline our method that scales DAPG to solve directly from visual information.

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. 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 onboard sensors like joint encoders (δ_t). To overcome the challenges associated with learning from high dimensional inputs, we use representations that project information into a lower-dimensional manifolds. These

---

Algorithm 1 RRL

1: **Input:** 25 Human Demonstrations ρ_D
2: **Initialize** using Behavior Cloning [Eq.4].
3: **repeat**
4:     **for** i = 1 to n **do**
5:         **for** t = 1 to horizon **do**
6:             Take action
            $a_t = \pi_\theta([Encoder(I_t), \delta_t])$
            and receive $I_{t+1}, \delta_{t+1}, r_{t+1}$
            from the environment.
7:         **end for**
8:     **end for**
9:     Compute $\nabla_\theta \log \pi_\theta(a_t|s_t)$ for each $(s,a) \in \rho_\pi, \rho_D$
10:    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, 8
11:    Calculate policy gradient according to 7
12:    Compute Fisher matrix 6
13:    Take the gradient ascent step according to 5.
14:    Update the parameters of the value function in order to approximate(2) : $V_k^\pi(s_t^{(n)}) \approx \sum_{t'=t}^T \gamma^{t'-t} r_t^{(n)}$
15: **until** Satisfactory performance

---

representations can be (a) learned in tandem with the RL objective. However, this leads to non-stationarity issue where the distribution induced by the current policy π_i may lie outside the expressive power of f, $\pi_i \not\subset \rho_i$ at any step i during training. (b) decoupled from RL by pre-training f. For this to work effectively, the feature extractor must be trained on a sufficiently wide distribution such that it covers any distribution that the policy might induce during training, $\pi_i \subset \rho \ \forall \ i$. Getting hold of such task specific training data beforehand becomes increasingly difficult as the complexity and diversity of the task increases. To this end, we propose to use a fixed feature extractor (Section 5.2) that is pretrained on a wide variety of real world scenarios like ImageNet dataset [Highlighted in *purple* in Figure 1]. We experimentally demonstrate that the diversity (Section 5.3) of the such feature extractor allows us to use it across all tasks we considered. The use of pre-trained representations induces stability to RRL as our representations are frozen and do-not face the non-stationarity issues encountered while learning policy and representation 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 (δ_t). As a substitute of the exact state (s_t), we empirically show that $[h_t, \delta_t]$ can be used as an input to the policy. In principle any RL algorithm can be deployed to learn the policy, in RRL we build upon Natural Policy Gradients (Kakade, 2001b) owing to effectiveness in solving complex high dimensional tasks (Rajeswaran et al., 2017). We present our full algorithm in Algorithm-1.

5. Experimental Evaluations

Our experimental evaluations aims to address the following questions: (1) Does pre-tained representations acquired via large real world image dataset allow RRL to learn complex tasks directly from proprioceptive signals (camera inputs and joint encoders)? (2) How does RRL's performance and efficiency compare against other state-of-the-art methods? (3) How various representational choices influence the generality and versatility of the resulting behaviors? (5) What are the effects of various design decisions on RRL? (6) Are commonly used benchmarks for studying image based continuous control methods effective?

5.1. Tasks

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 like Cartpole, Cheetah, Reacher, Finger spin, Walker, Ball in cup, etc. Moving beyond these simple domains, we investigate RRL on Adroit manipulation suite (Rajeswaran et al., 2017) which consists of contact-rich high-dimensional dexterous manipulation tasks (Figure 3) that have found to be challenging ever for state (s_t) based methods. Furthermore, unlike prior task sets, which are fundamentally planar and devoid of depth perspective, the Adroit manipulation suite consists of visually-rich physically-realistic tasks that demand representations untangling complex depth information.

5.2. Implementation 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 of the model is removed to recover a 512 dimensional feature space 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 7). All the hyperparameters used for training are summarized in Appendix(Table 2). We report an average performance over three random seeds for all the experiments.

5.3. Results

In Figure 4, we contrast the performance of RRL against the state of the art baselines. We begin by observing that NPG (Kakade, 2001b) 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 and a few demonstrations from
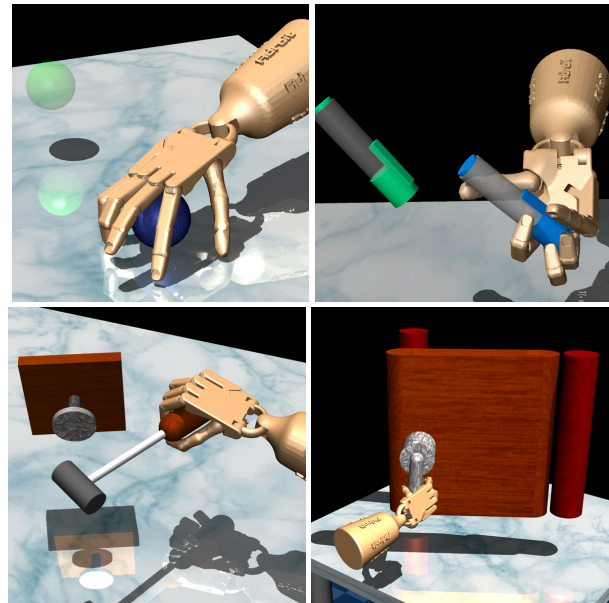

Figure 3. ADROIT manipulation suite consisting of complex dexterous manipulation tasks involving object relocation, in hand manipulation (pen repositioning), tool use (hammering a nail), and interacting with human centric environments (opening a door).

the environment to solve the tasks and pose as the best case oracle. RRL demonstrates good performance on all the tasks, relocate being the hardest, and often approaches performance comparable to our strongest oracle-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. In Figure 5 [Left] we compare the computational footprint of FERM (along with other methods, discussed in later sections) with RRL. We note that our method not only outperforms FERM but also is approximately five times more compute-efficient.

5.4. Effects of Visual Distractors

In Figure 5 [Center, Right] we probe the robustness of the final policies by injecting visual distractors in the environment during inference. We note that the resilience of the resnet features induces robustness to RRL's policies. On the other hand, task-specific features learned by FERM are brittle leading to larger degradation in performance. In addition to improved sample and time 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, clutter in the scene. More details about the experiment setting is provided in Section 7.8 in Appendix.

---

[1]Reporting best performance amongst over 30 configurations per task we tried in consultation with the FERM authors.
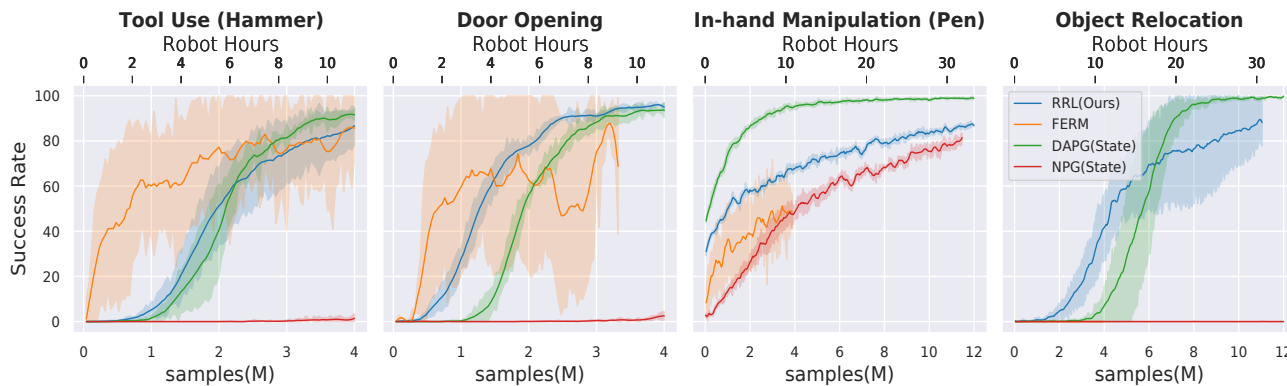
Figure 4. Performance on ADROIT dexterous manipulation suite (Rajeswaran et al., 2017): State of the art policy gradient method NPG(State) (Rajeswaran et al., 2018) struggles to solve the suite even with privileged low level state information, establishing the difficulty of the suite. Amongst demonstration 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).
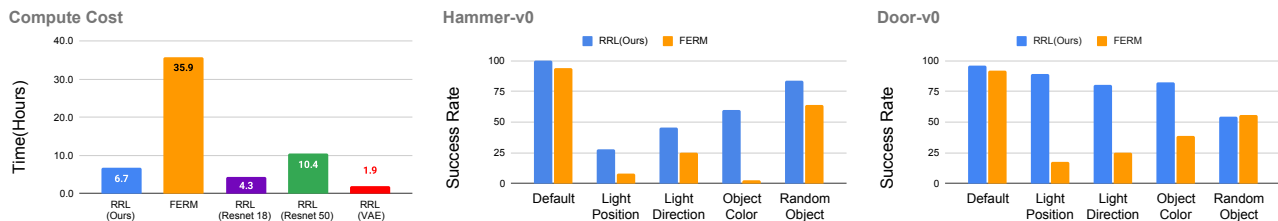

Figure 5. LEFT: Comparison of the computational cost of RRL with Resnet34 i.e RRL(Ours), FERM - Strongest baseline, RRL with Resnet 18, RRL with Resnet 50 and RRL(VAE) baseline. CENTER,RIGHT: Influence of various environment distractions (lightning condition, object color) on RRL(Ours), and FERM. RRL(Ours) consistently performs better than FERM in all the variations we considered.

5.5. Effect of Representation

Is Resnet lucky? To investigate if architectural choice of Resnet is lucky, in Figure 6 we test different models pre-trained on ImageNet dataset as RRL's feature extractors – MobileNetV2 (Sandler et al., 2019), ShuffleNet (Ma et al., 2018) and state of the art hierarchical VAE (Child, 2021) [Refer Section 7.5 in Appendix for more details]. Not much degradation in performance is observed with respect to the Resnet model. This highlights that it is not the architecture choices in particular, rather the dataset on which models are being pre-trained, that delivers generic features effective for the RL agents.

Task-specific vs Task-agnostic representation: In Figure 7, we compare the performance between (a) learning task specific representations (VAE) (b) generic representation trained on a very wide distribution (Resnet). We note that RRL using Resnet34 significantly outperforms a variant RRL(VAE) (see appendix for details Section 7.7) that learns features via commonly used variational inference techniques on a task specific dataset (Ha & Schmidhuber, 2018a;b; Higgins et al., 2018; Nair et al., 2018). This indicates that pre-trained Resnet provides task agnostic and superior features compared to methods that explicitly learn
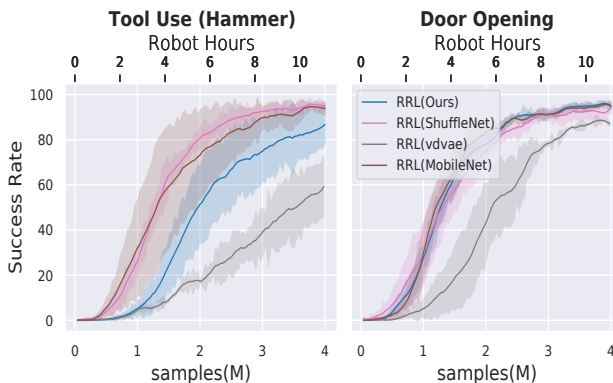

Figure 6. Effect of different types of Feature extractor pretrained on ImageNet dataset, highlighting that not just Resnet but any feature extractor pretrained on a sufficiently wide distribution of data remains effective.

brittle (Section-5.8) and task-specific features using additional samples from the environment. It is important to note that the latent dimension of the Resnet34 and VAE are kept same (512) for a fair comparison, however, the model sizes are different as one operates on a very wide distribution while the other on a much narrower task specific dataset. Additionally, we summarize the compute cost of
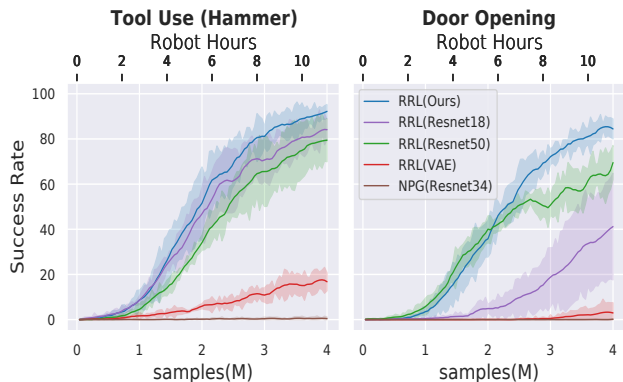
Figure 7. Influence of representation: RRL(Ours), using resnet34 features, outperforms commonly used representation (RRL(VAE)) learning method VAE Amongst different Resnet variations, Resnet34 strikes the balance between representation capacity and computational overhead. NPG(Resnet34) showcases the performance with Resnet34 features but without demonstration bootstrapping, indicating that only representational choices are not enough to solve the task suite.

both the methods RRL(Ours) and RRL(VAE) in Figrue 5 [Left]. We notice that even though RRL(VAE) is the cheapest, its performance is quite low (Figure 7). RRL(Ours) strikes a balance between compute and efficiency.

5.6. Effects of proprioception choices and sensor noise
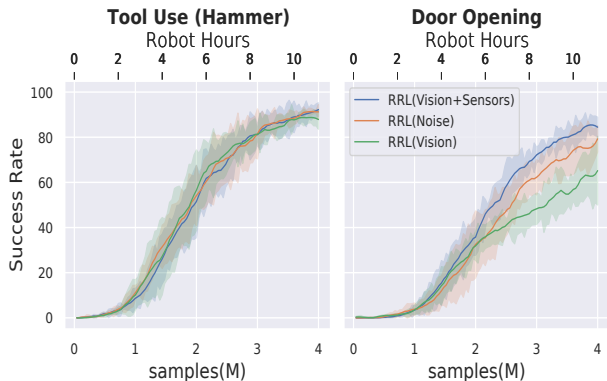

Figure 8. Influence of proprioceptive signals on RRL(Vision+sensors-Ours): RRL(Noise) demonstrates that RRL remains effectiveness in presence of noisy (2%) proprioception. RRL(Vision) demonstrates that RRL remains performant with (only) visual inputs as well.

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 8, we subjected RRL to (a) signals with 2% noise in the information received from the joint encoders RRL(Noise), and (b) only visual inputs are used as proprioceptive signals RRL(Vision). In both these cases, our methods remained performant with slight to no degradation in performance.
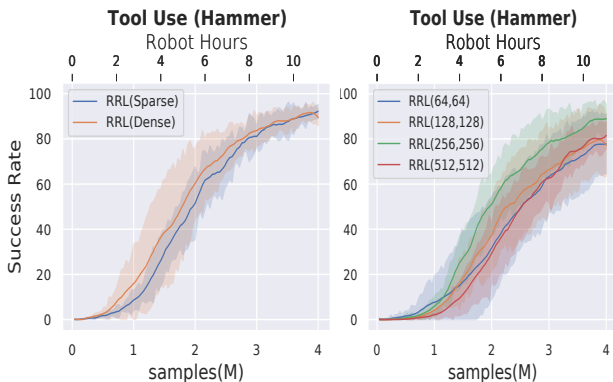

Figure 9. LEFT: Influence of rewards signals: RRL(Ours), using sparse rewards, remains performant with a variation RRL_{dense} using well-shaped dense rewards. RIGHT: Effect of policy size on the performance of RRL . We observe that it is quite stable with respect to a wide range of policy sizes.

5.7. 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 7, we study the effect of different Resnet features as our representation. Resnet34, though computationally more demanding (Figure 5) 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.

Reward design, especially for complex high dimensional tasks, requires 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 expertise required for our methods. In Figure 9-LEFT, we observe that RRL (using sparse rewards) delivers competitive performance to a variant of our methods that uses well-shaped dense rewards while being resilient to variation in policy network capacity (Figure 9-RIGHT).

5.8. Rethinking benchmarking for visual RL

DMControl (Tassa et al., 2018) is a widely used benchmark for proprioception based RL methods – RAD (Laskin et al., 2020), SAC+AE (Yarats et al., 2020), CURL (Srinivas et al., 2020), DrQ (Kostrikov et al., 2020). While these methods perform well (Table 1) on such simple DMControl tasks, their progress struggles to scale when met with task representative of real world complexities such as realistic Adroit Manipulation benchmarks (Figure 4).

For example we demonstrate in Figure 4 that a representative SOTA methods FERM (uses expert demos along with

500K Step Scores	RRL+SAC	RAD	Fixed RAD Encoder	CURL	SAC+AE	State SAC
Finger, Spin	422 ± 102	$\mathbf{947} \pm 101$	789 ± 190	926 ± 45	884 ± 128	923 ± 211
Cartpole, Swing	357 ± 85	863 ± 9	$\mathbf{875} \pm 01$	845 ± 45	735 ± 63	848 ± 15
Reacher, Easy	382 ± 299	$\mathbf{955} \pm 71$	53 ± 44	929 ± 44	627 ± 58	923 ± 24
Cheetah, Run	154 ± 23	$\mathbf{728} \pm 71$	203 ± 31	518 ± 28	550 ± 34	795 ± 30
Walker, Walk	148 ± 12	$\mathbf{918} \pm 16$	182 ± 40	902 ± 43	847 ± 48	948 ± 54
Cup, Catch	447 ± 132	$\mathbf{974} \pm 12$	719 ± 70	959 ± 27	794 ± 58	974 ± 33
100K Step Scores						
Finger, Spin	135 ± 67	$\mathbf{856} \pm 73$	655 ± 104	767 ± 56	740 ± 64	811 ± 46
Cartpole, Swing	192 ± 19	828 ± 27	$\mathbf{840} \pm 34$	582 ± 146	311 ± 11	835 ± 22
Reacher, Easy	322 ± 285	$\mathbf{826} \pm 219$	162 ± 40	538 ± 233	274 ± 14	746 ± 25
Cheetah, Run	72 ± 63	$\mathbf{447} \pm 88$	188 ± 20	299 ± 48	267 ± 24	616 ± 18
Walker, Walk	63 ± 07	$\mathbf{504} \pm 191$	106 ± 11	403 ± 24	394 ± 22	891 ± 82
Cup, Catch	261 ± 57	$\mathbf{840} \pm 179$	533 ± 148	769 ± 43	391 ± 82	746 ± 91

Table 1. **Results on DMControl Benchmark.** RAD outperforms all the baselines whereas RRL performs worse in the 100K and 500k Environmental step benchmark suggesting that it is quicker to learn task specific representation in simple tasks whereas Fixed Rad Encoder highlights that the representations learned by RAD are narrow and task specific.

RAD) struggles to perform well on Adroit Manipulation benchmark. On the contrary, RRL using Resnet features pretrained on real world image dataset, delivers state comparable results on Adroit Manipulation benchmark while struggles on the DMControl (RRL+SAC: RRL using SAC and Resnet34 features 1). This highlights large domain gap between the DMControl suite and the real-world.

We further note that the pretrained features learned by SOTA methods aren't as widely applicable. We use a pre-trained RAD encoder (pretrained on Cartpole) as fixed feature extractor (Fixed RAD encoder in Table 1) and retrain the policy using these features for all environments. The performance degrades for all the tasks except Cartpole. This highlights that the representation learned by RAD (even with various image augmentations) are task specific and fail to generalize to other tasks set with similar visuals. Furthermore, learning such task specific representations are easier on simpler scenes but their complexity grows drastically as the complexity of tasks and scenes increases. To ensure that important problems aren't overlooked, we emphasise the need for the community to move towards benchmarks representative of realistic real world tasks.

6. Strengths, Limitations & Opportunities

This paper presents an intuitive idea bringing together advancements from the fields of representation learning, imitation learning, and reinforcement learning. We present 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.

Strengths: The strength of our insight lies in its simplicity, and applicability to almost any reinforcement or imitation

learning algorithm that intends to learn directly from high dimensional proprioceptive signals. We present RRL, an instantiation of this insight on top of imitation + (on-policy) reinforcement learning methods called DAPG, to showcase its strength. It presents yet another demonstration that features learned by Resnet are quite general and are broadly applicable. Resnet features trained over 1000s of real-world images are more robust and resilient in comparison to the features learned by methods that learn representation and policies in tandem using only samples from the task distribution. The use of such general but frozen representations in conjunction with RL pipelines additionally avoids the non-stationary issues faced by competing methods that simultaneously optimizes reinforcement and representation objectives, leading to more stable algorithms. Additionally, not having to train your own features extractors results in a significant sample and compute gains, Refer to Figure 5.

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 all the 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 leveraging data from various sources, building hardware-accelerated devices using feature compression, low latency and low bandwidth 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.

Child, R. Very deep vaes generalize autoregressive models and can outperform them on images, 2021.

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. A natural policy gradient. In *NIPS*, 2001a.

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

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.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design, 2018.

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.

Rajeswaran, A., Mordatch, I., and Kumar, V. A game theoretic framework for model based reinforcement learning, 2020.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

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.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. Deepmind control suite, 2018.

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.

Yarats, D. and Kostrikov, I. Soft actor-critic (sac) implementation in pytorch. https://github.com/denisyarats/pytorch_sac, 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.