Multi-Task Reinforcement Learning without Interference

Tianhe Yu[1], **Saurabh Kumar**[1], **Abhishek Gupta**[2], **Sergey Levine**[2], **Karol Hausman**[3], **Chelsea Finn**[1]
Stanford University[1], UC Berkeley[2], Robotics at Google[3]
tianheyu@cs.stanford.edu

Abstract

While deep reinforcement learning systems have demonstrated impressive results in domains ranging from game playing and robotic control, sample efficiency remains a major challenge, particularly as these algorithms learn individual tasks from scratch. Multi-task and goal-conditioned reinforcement learning have emerged as promising approaches for sharing structure across multiple tasks to enable more efficient learning. However, challenges in optimization have hamstrung such methods from realizing efficiency gains compared to learning tasks independently from scratch. Motivated by these challenges, we develop a general approach that can change the multi-task optimization landscape to alleviate conflicting gradients across tasks. In particular, we introduce two instantiations of this approach, one architectural and one algorithmic, that prevent gradients for different tasks from interfering with one another. On two challenging multi-task RL problems, we find that our approaches leads to greater final performance and learning efficiency in comparison to prior approaches.

1 Introduction

While reinforcement learning (RL) is a promising approach for enabling agents to autonomously acquire behaviors, the data requirements of current methods preclude the ability to learn a breadth of behaviors, if all tasks are learned individually from scratch. A natural approach to such multi-task RL problem settings is to train on all tasks jointly, with the aim of using shared structure to achieve greater efficiency and performance than solving tasks individually. However, learning multiple tasks all at once has led to significant challenges in optimization (Parisotto et al., 2015; Rusu et al., 2015), often leading to *worse* asymptotic performance and sample efficiency compared to learning tasks individually, particularly when aiming to learn a diverse set of discrete tasks. In this work, we aim to study whether such challenges can be overcome with simple changes to the architecture or learning algorithm.

We hypothesize that difficulties in multi-task optimization are caused by interference between the gradients of multiple tasks (Schaul et al., 2019). To address this issue, prior works have attempted to learn tasks in independent networks and then distill these networks into a single policy (Levine et al., 2016; Parisotto et al., 2015; Rusu et al., 2015; Ghosh et al., 2017; Teh et al., 2017). While this can alleviate optimization issues, these approaches have been most successful in settings where different tasks correspond to continuously varying goals within the same underlying problem. Further, because these approaches decouple the optimization problem by task, they are often unable to acquire efficiency gains by recognizing the shared structure, especially when considering multiple distinct tasks. Instead, we hope to develop a simple yet general approach for preventing gradient interference while still allowing for positive transfer.

In this work, we find that simple changes to the architecture or gradient application can prevent these optimization challenges entirely. Geometrically, we consider interference to be caused by gradients

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

with a negative cosine similarity, and our key insight is to prevent the interfering components of the gradient from being applied to the learned network. This can be accomplished both with a simple architectural change, by rotating the activations at each layer in a learned and differentiable way per task, and in an algorithmic way, by altering the gradients for each task.

The key contribution of this work is a multi-task reinforcement learning algorithm that avoids gradient interference and, as a result, can scale to large sets of discrete tasks. Our approach for deconflicting gradients (D-Grad) requires only a single modification to either the architecture or the application of gradients, and hence can easily be combined with existing reinforcement learning algorithms. We combine our approach with the soft actor-critic method (Haarnoja et al., 2018b), and evaluate it on two challenging sets of distinct manipulation tasks in the Meta-World benchmark (Yu et al., 2019). In our evaluation, we find that D-Grad leads to significant improvements in terms of both learning efficiency and performance compared to joint multi-task training.

2 Related Work

Algorithms for multi-task learning typically consider how to train a single model that can solve a variety of different tasks (Caruana, 1997; Bakker & Heskes, 2003; Espeholt et al., 2018). Many multi-task RL approaches aim to decompose the problem into multiple local problems, e.g. each task, that are significantly easier to learn akin to divide and conquer algorithms (Levine et al., 2016; Teh et al., 2017; Ghosh et al., 2017; Rusu et al., 2016a; Czarnecki et al., 2019; Parisotto et al., 2015). Eventually, the local models are combined into a single, multi-task policy using different distillation techniques (outlined in (Czarnecki et al., 2019)). While these algorithms tend to perform well in the context of multi-goal RL, they limit the potential benefits of multi-task learning such as sharing data between the tasks (Riedmiller et al., 2018; Andrychowicz et al., 2017) or small memory footprint, and can suffer from challenges associated with supervised policy learning such as compounding errors (Ross et al., 2011). As a result, these methods have been most successful in settings where the tasks share considerable structure. In contrast, we propose a simple and cogent scheme for multi-task RL that retains many of the benefits while not requiring the additional memory and complex machinery required for distillation-style algorithms.

Approached from an alternative perspective, a number of more architectural solutions have been proposed to the multi-task RL problem based on multiple modules or paths (Fernando et al., 2017; Devin et al., 2016; Misra et al., 2016; Rusu et al., 2016b; Rosenbaum et al., 2017; Vandenhende et al., 2019; Wulfmeier et al., 2019), using attention-based architectures (Liu et al., 2018; Maninis et al., 2019), or using the idea of network superposition (Cheung et al., 2019). Another set of approaches (Hausman et al., 2018; Tirumala et al., 2019; Heess et al., 2016; Haarnoja et al., 2018a) focus on the right task representation that simplifies learning multi-task policies, which can be later utilized as a part of a hierarchical policy. In contrast to these works, our work focuses on the conflicting gradients aspect of multi-task RL, and we propose simple architectural and optimization changes to address this problem directly.

Similarly to our work, a number of prior approaches have observed the difficulty of optimization in the multi-task learning setting (Hessel et al., 2019; Sener & Koltun, 2018; Chen et al., 2018; Schaul et al., 2019). Schaul et al. (2019) studies the challenges that arise in multi-task RL from performing multi-objective optimization in a setting where learning and data generation are coupled, and the learned behavior controls the future data distribution. This work attributes the problem to the concept of "ray interference", which is related to the conflicting gradients problem, but the work does not provide solutions to this challenge. Alternatively, Chen et al. (2018); Kendall et al. (2018) attribute the challenges of multi-task learning to the imbalance between gradient magnitudes across different tasks and propose an adaptive gradient normalization to account for it. Hessel et al. (2019) considers a similar insight in the case of reinforcement learning. Building on this, Sener & Koltun (2018) aims to find a pareto-optimal solution to the multi-objective optimization rather than trying to optimize a weighted sum of the multi-task objectives. Our work, in contrast to many of these optimization schemes, suggests that the challenge in multi-task learning can be attributed to the problem of conflicting gradients, which we can address directly by introducing practical, gradient-deconflicting algorithms.

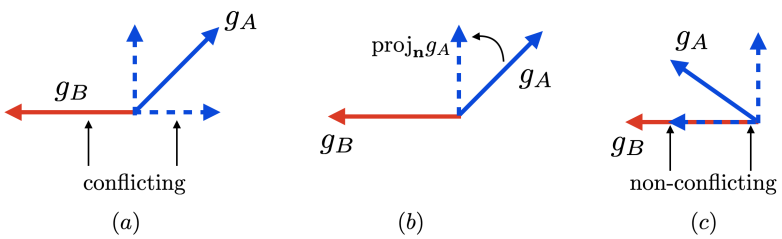


Figure 1: Visual depiction of conflicting gradients phenomenon. In (a), we see that tasks A and B have conflicting gradient directions, which can lead to destructive interference and unstable learning. In (b), we illustrate the D-Grad (O) algorithm in cases where gradients are conflicting - we project the gradient onto the normal plane. In (c), we show that tasks with non-conflicting gradients are not altered under D-Grad (O), thereby keeping tasks with constructive interference.

3 Preliminaries

We utilize the formalism of finite-horizon Markov Decision Processes (MDPs) where each task is defined by its own MDP $\mathcal{M} = (S, A, P, R, H, \gamma)$, where $s \in S$ correspond to states, $a \in A$ correspond to the available actions, $P(s_{t+1}|s_t, a_t)$ represents the stochastic transition dynamics, $R(s, a)$ is a reward function, H is the horizon and γ is the discount factor.

While the goal in single-task reinforcement learning is to learn a policy $\pi(a|s)$ that maximizes the sum of discounted task-specific rewards: $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R_t(s_t, a_t)]$, also known as the return, multi-task reinforcement learning assumes access to a distribution of tasks $p(\mathcal{T})$ and aims to find a policy that maximizes the expected return across all tasks drawn from $p(\mathcal{T})$: $\mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}[\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R_t(s_t, a_t)]]$. In order to obtain a policy that solves a specific task from the task distribution $p(\mathcal{T})$, multi-task reinforcement learning defines a task-conditioned policy $\pi(a|s, z)$ where z is an encoding of the task ID, which could be provided as a one-hot vector or in any other, unique form. Multi-task reinforcement learning algorithms are commonly evaluated based on their average performance over the training tasks.

4 Multi-Task Reinforcement Learning without Conflict

While the multi-task problem can in principle be solved by simply applying a standard single-task RL algorithm with a suitable goal identifier provided to the policy, a number of prior works (Ghosh et al., 2017; Teh et al., 2017) have found this learning problem to be extremely difficult in the reinforcement learning setting. In this work, we hypothesize that one of the main challenges of multi-task RL can be characterized as conflicting or thrashing gradients and find that it can significantly impede learning progress when combined with iterative data collection. We identify possible causes for this problem, and propose two different algorithms to mitigate it in the multitask reinforcement learning setting.

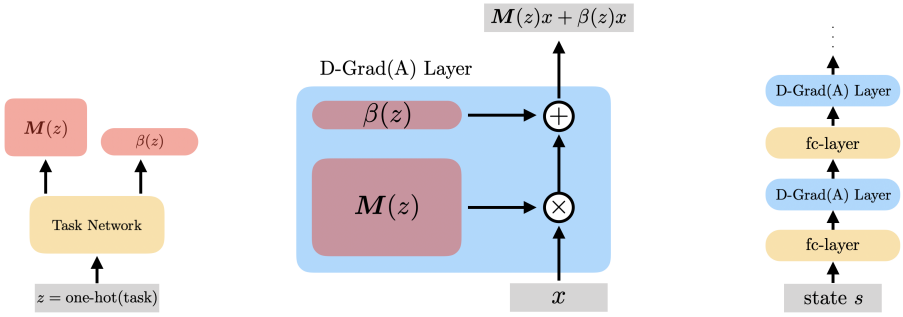
4.1 Avoiding Conflicting Gradients

The phenomenon of conflicting and thrashing gradients can be easily understood pictorially as shown in Fig 1. When multiple tasks are trained simultaneously with shared parameters, then the gradients which are being applied from different tasks will conflict with each other when their cosine similarity is negative. When the sum of these gradients is applied to the parameters in a gradient descent step, the resulting step may be highly suboptimal, since the gradient for one task effectively cancels the other one out. Essentially, if the gradients across tasks for the current parameters are opposing, the application of those gradients may lead to minimal learning progress since the gradients cancel each other out. Empirically, we see that this manifests through the process of thrashing gradients where the gradients don't cancel each other out but instead change direction very frequently across batches, leading to a system where learning is very difficult since the learning direction is constantly changing.

How do we devise learning algorithms which can learn in multi-task settings without suffering from the problems associated with thrashing and conflicting gradients? Our goal is to *deconflict* the task-specific gradients throughout the learning process so as to limit the negative interference between the tasks. One approach for doing this is to introduce architectural changes to our models,

which make the issue of conflicting gradients less prevalent, while retaining the same optimization process. Alternatively, we can change the optimization process to directly eliminate the problem of conflicting and thrashing gradients, while retaining standard network architectures. We propose two algorithms based on these principles - deconflicting gradients via architecture (DGrad-A) and deconflicting gradients via optimization (DGrad-O).

4.2 Architectural Solution: D-Grad via Architecture



(a) (b) (c)

Figure 2: D-Grad (A) architecture. (a) A task network receives a one-hot task vector z as input and produces a matrix $M(z)$ and vector $\beta(z)$. (b) A D-Grad(A) Layer applies a matrix multiplication and vector addition to an input vector x. The matrix and vector used to linearly transform the input vector are given by a task network. (c) The modified RL policy or critic model with a D-Grad(A) layer added after each fully-connected layer.

To mitigate the problem of conflicting gradients during optimization, we can employ a few very simple architectural changes in the neural network model being learned to encourage the model to have orthogonal gradients. Intuitively, we simply introduce a particular variant of multiplicative interactions with network activations, which greatly reduces the occurrence of conflicting gradients during optimization.

Let us consider the optimization of a task-conditioned policy $\pi(a|s, z)$, where z is a one-hot vector denoting the task and s, a are the current state and action respectively. Given this setup, and some intermediate layer l of the network, with associated activations x, we introduce a transformation via multiplicative interactions which encourages the activations of each layer of the network to be orthogonal to the layer's activations when conditioned on other task vectors. Associated with each layer l of the network, we randomly initialize a task network which takes as input z and outputs a matrix $\boldsymbol{M}_l(z)$ and a vector $\beta_l(z)$. We then transform the activations of l using a *D-Grad A* layer, as defined below.

Definition 1. *A D-Grad (A) layer applies the following transformation to an activation vector* $x \in \mathbb{R}^n$*:*

$$\textbf{\textit{D-Grad A}}(x) = \boldsymbol{M}(z)x + \beta(z) \tag{1}$$

where $\boldsymbol{M}(z) \in \mathbb{R}^{n \times n}$ *is an* z*-dependent matrix and* $\beta(z) \in \mathbb{R}^n$ *is a* z*-dependent vector.*

While at first glance, this simple change seems to be very similar to FiLM (Perez et al., 2018), it is subtly different in the way the conditioning layer interacts with the network. While FiLM interacts elementwise with activations per-layer, the D-Grad (A) employs a matrix multiplication which allows for a more expressive rotation via the conditioning layer. We find empirically that using D-Grad (A) provides significant gains over baseline variants of SAC, since it allows the gradients to deconflict from each other for different tasks.

4.3 Algorithmic Solution: D-Grad via Optimization

The method described in Section 4.2, considers how to modify the model architecture to minimize the chance of gradient conflict across tasks. While this simple proposal should allow us to partly address the conflicting gradients problem, we demonstrate an alternative method to directly modify the optimization so as to deconflict gradients across different tasks. The key insight we leverage to here

Algorithm 1 DGrad-O algorithm

for $task_i = A, B, C..K$ **do**
  Compute gradient g_i of $task_i$
  for $task_j = A, B, C, ...K \setminus task_i$ **do**
    Compute gradient g_j of task j.
    Compute project direction of g_i onto g_j as $d = \frac{g_i \cdot g_j}{\|g_j\| \|g_j\|}$.
    if $d < 0$ **then**
      Set $g_i = g_i - \frac{g_j}{\|g_j\|} \frac{g_i \cdot g_j}{\|g_j\|}$ *// Subtract the projection of g_i onto g_j*
    end if
  end for
  Store $g^{\text{proj}}_{task_i} = g_i$
end for
Apply combined sum of gradients across tasks $\theta^{'} = \theta - \alpha \sum_{\tau=A,B,C,D,...} \left[g^{\text{proj}}_{\tau} \right]$

is to note that we can characterize tasks as conflicting by noting the cosine similarity between their respective gradient directions. In particular, negative cosine similarity between the gradient vectors indicates that the tasks are likely to be interfering negatively, while the positive cosine similarity indicates constructive interference.

Since the thrashing gradients issue is largely caused by negative interference, we can adopt a simple procedure to deconflict gradients during optimization - if the gradients between tasks are in conflict, we simply project the gradient along the normal plane of the gradient of the other task. This amounts to removing the conflicting part of the gradient for the task, thereby reducing the amount of destructive gradient interference between tasks. Suppose the gradient for task A is g_A, and the gradient for task B is g_B, then we can deconflict gradients as follows - (1) First determine whether g_A conflicts with g_B. We can determine this by computing the cosine similarity between vectors g_A and g_B: negative values indicating conflicting gradients. (2) In case the gradients are conflicting, replace g_A by its projection onto the normal plane of g_B: $g_A = g_A - \frac{g_B}{\|g_b\|} \frac{g_A \cdot g_B}{\|g_B\|}$. If the gradients are non-conflicting, then we simply leave g_A as is. (3) We can then repeat this with all the other tasks sampled in the current batch $C, D, ...$ to get the eventual gradient g^{proj}_A to apply for task A. We can then use the same procedure for the other tasks in the batch $B, C, D...$ to get the applicable gradients g^{proj}_B, g^{proj}_C and so on. These can then be applied to the parameters via a descent step, with learning rate α.

$$\theta^{'} = \theta - \alpha \sum_{\tau=A,B,C,D,...} \left[g^{\text{proj}}_{\tau} \right] \tag{2}$$

This procedure, while simple to implement, ensures that the gradients which we apply for each task per batch are minimally interfering with the other tasks in the batch, mitigating the conflicting gradient problem to a large extent. The algorithm is described in detail in Algorithm 1. Our experimental results verify the hypothesis that this procedure boosts learning progress, and significantly reduces the problem of conflicting gradients while learning. A pictoral description can be found in Fig 1.

While this procedure can be applied to any gradient-based reinforcement learning algorithm, we instantiate it to be applied with an off-policy actor-critic algorithm (soft actor-critic by Haarnoja et al. (2018b)). In SAC, we apply D-Grad (O) to the gradient updates for both the actor and the critic, as described further in our experimental evaluation.

5 Experiments

The goal of our experimental evaluation is to answer the following question: how does D-Grad compare to prior approaches to multi-task learning when solving distinct task families?

To answer these questions, we evaluate D-Grad on two sets of manipulation tasks, MT10 and MT50, from the Meta-World benchmark (Yu et al., 2019), shown in Figures 5 and 4. The two task sets are challenging for multi-task RL algorithms to tackle as the agent is required to master a variety of distinct skills such as reaching and grasping to solve all tasks. We train D-Grad using SAC and compare it to a number of methods on MT10 and MT50:

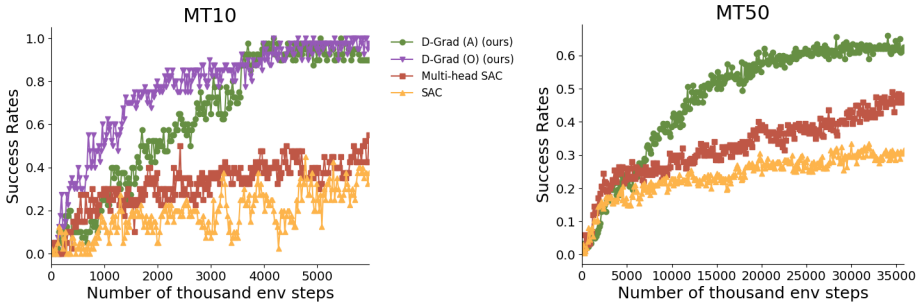


Figure 3: Learning curves on MT10 (left) and MT50 (right). The two D-Grad variants outperform the other methods in the two settings in terms of both success rates and data efficiency.

- **SAC**: the vanilla SAC algorithm conditioned on the task one-hot vectors.
- **Multi-head SAC**: Both the policy and Q networks of SAC are multi-head feedforward neural networks where each head output the result of each task.

We represent D-Grad (A) as 3-layer fully-connected feedforward neural networks with 160 hidden units for both the actor and the critic. Since we apply a matrix-vector multiplication after the activation of each layer, we use a 6-layer fully-connected feedforward neural networks with 160 hidden units for D-Grad (O) and all the three baselines such that they enjoy about the same expressiveness as that of D-Grad (A). For D-Grad (O), we apply the deconflicting gradient procedure for both the critic and actor networks. We adopt the default hyperparameters from SAC for training all the methods.

5.1 MT10 evaluation

For this evaluation, we test all methods on a subset of the 50 manipulation tasks shown in Figure 5. Specifically, we conduct comparisons on 10 tasks shown in Figure 4. The results are shown in Figure 3 on the left. The success rates are averaged across tasks and we adopt the success metrics used in the Meta-World benchmark. D-Grad (O) learns all tasks with the best data efficiency, while D-Grad (A) also successfully solves all 10 tasks in 3 million environment steps. Training a single SAC policy and a multi-head policy turns out to be unable to acquire half of the skills, suggesting that eliminating gradient interference across tasks can significantly boost performance of multi-task RL.

As noted in Yu et al. (2019), these tasks involve fairly distinct behavior motions, which makes learning all of them with a single policy quite challenging. The ability to learn these tasks together opens the door for a number of interesting extensions to meta-learning, goal conditioned RL and generalization to novel task families.

5.2 MT50 evaluation

We conduct a more challenging experiment where all methods are evaluated on all 50 tasks from Meta-World. In this hard setting, as shown on the right in Figure 3, D-Grad (A) quickly learns to solve more than 60% of tasks in 20 million environment steps while SAC and SAC with multi-head architectures struggled in solve 40% of the tasks after 35 million steps. This result demonstrates that D-Grad leads to notable improvement in challenging multi-task RL domains.

6 Conclusion

In this paper, we presented an approach mitigating optimization issues in multi-task reinforcement learning by deconflicting gradients. We introduced two instantiations of this approach using changes to the architecture or to the gradient application. Our results indicate that, on a challenging set of 10 and 50 distinct tasks respectively, our approach learns significantly faster and reaches significantly greater final performance in comparison to training networks jointly. This paper opens up several interesting directions for future investigation. In particular, the ideas described in the paper are not specific to multi-task reinforcement learning, and can also, in principle, be applied to multi-task supervised learning, goal-conditioned reinforcement learning, and meta-learning problem settings.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *J. Mach. Learn. Res.*, 2003.

Rich Caruana. Multitask learning. *Machine Learning*, 1997.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the International Conference on Machine Learning*, 2018.

Brian Cheung, Alex Terekhov, Yubei Chen, Pulkit Agrawal, and Bruno A. Olshausen. Superposition of many models into one. *CoRR*, abs/1902.05522, 2019.

Wojciech M. Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant M. Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*, 2019.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. *CoRR*, abs/1609.07088, 2016.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734, 2017. URL `http://arxiv.org/abs/1701.08734`.

Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. *CoRR*, abs/1711.09874, 2017.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. *arXiv preprint arXiv:1804.02808*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning*, 2018b.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. 2018.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3796–3803, 2019.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. *CoRR*, abs/1803.10704, 2018.

Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. *CoRR*, abs/1904.08918, 2019.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *CoRR*, abs/1711.01239, 2017.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *4th International Conference on Learning Representations, ICLR*, 2016a.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016b.

Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. Ray interference: a source of plateaus in deep reinforcement learning. *arXiv preprint arXiv:1904.11455*, 2019.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 527–538, 2018.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

Dhruva Tirumala, Hyeonwoo Noh, Alexandre Galashov, Leonard Hasenclever, Arun Ahuja, Greg Wayne, Razvan Pascanu, Yee Whye Teh, and Nicolas Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019.

Simon Vandenhende, Bert De Brabandere, and Luc Van Gool. Branched multi-task networks: Deciding what layers to share. *CoRR*, abs/1904.02920, 2019.

Markus Wulfmeier, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Tim Hertweck, Thomas Lampe, Noah Siegel, Nicolas Heess, and Martin Riedmiller. Regularized hierarchical policies for compositional transfer in robotics. *arXiv preprint arXiv:1906.11228*, 2019.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Sergey Levine, and Chelsea Finn. Meta-world: A benchmark and evaluation for multi-task and meta-reinforcement learning, 2019. URL `https://github.com/rlworkgroup/metaworld`.

A Appendix

We show the visualizations of all tasks evaluated in MT10 and MT50 in Figure 4 and Figure 5 respectively, which demonstrates the diversity of tasks that we evaluate methods on.

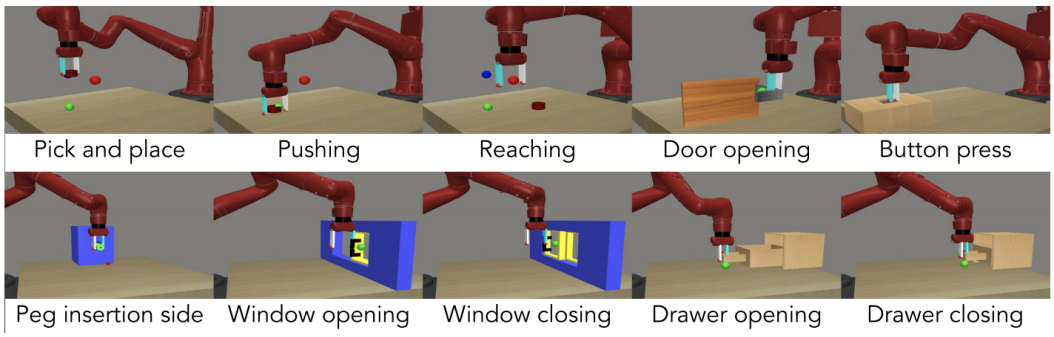


Figure 4: Visualization of 10 tasks used in Meta-World MT10 evaluation.

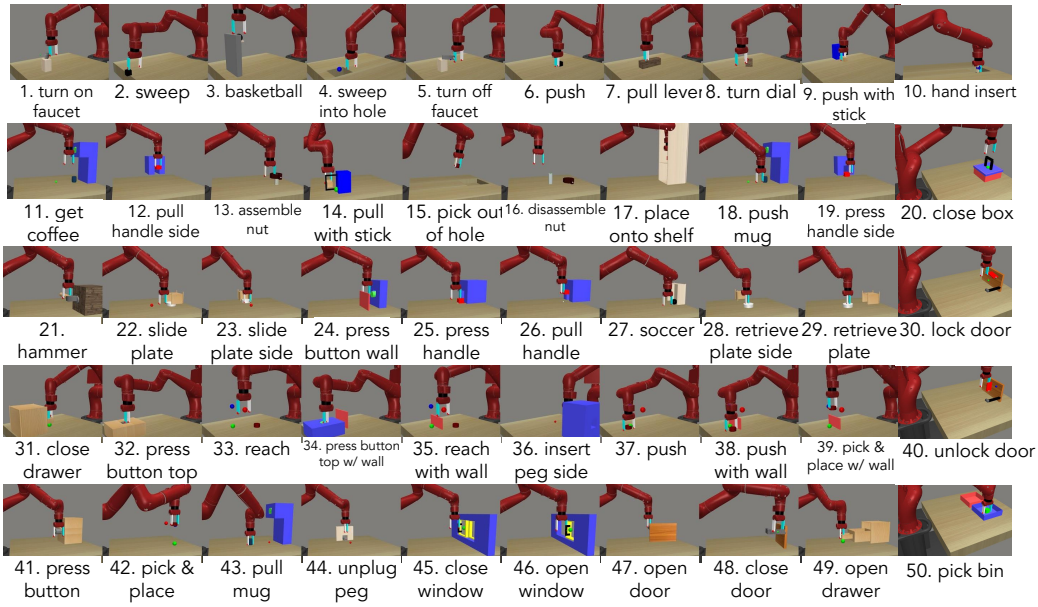


Figure 5: Visualization of the 50 tasks from Meta-World used in the MT50 evaluation.