Learning Task-Distribution Reward Shaping with Meta-Learning

Haosheng Zou[1], **Tongzheng Ren**[2], **Dong Yan**[1], **Hang Su**[1], **Jun Zhu**[1]*
[1]Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Lab, Bosch-Tsinghua Joint ML Center, Tsinghua University
[2]Department of Computer Science, UT Austin
zouhaosheng@163.com, {rtz19970824, sproblvem}@gmail.com, {suhangss, dcszj}@tsinghua.edu.cn

Abstract

Reward shaping is one of the most effective methods to tackle the crucial yet challenging problem of credit assignment and accelerate Reinforcement Learning. However, designing shaping functions usually requires rich expert knowledge and hand-engineering, and the difficulties are further exacerbated given multiple tasks to solve. In this paper, we consider reward shaping on a *distribution* of tasks that share state spaces but not necessarily action spaces. We provide insights into optimal reward shaping, and propose a novel meta-learning framework to automatically *learn* such reward shaping to apply on newly sampled tasks. Theoretical analysis and extensive experiments establish us as the state-of-the-art in learning *task-distribution reward shaping*, outperforming previous such works (Konidaris and Barto 2006; Snel and Whiteson 2014). We further show that our method outperforms learning intrinsic rewards (Yang et al. 2019; Zheng et al. 2020), outperforms Rainbow (Hessel et al. 2018) in complex pixel-based CoinRun games, and is also better than hand-designed reward shaping on grid mazes. While the goal of this paper is to learn reward shaping rather than to propose new general meta-learning algorithms as PEARL (Rakelly et al. 2019) or MQL (Fakoor et al. 2020), our framework based on MAML (Finn, Abbeel, and Levine 2017) also outperforms PEARL / MQL, and could combine with them for further improvement.

1 Introduction

Credit assignment (Minsky 1961) remains a key challenge on the *learning efficiency* (sample complexity) of Reinforcement Learning (RL) (Mnih et al. 2015; Silver et al. 2018), which usually requires many samples from the environment to learn well. *Reward shaping* is one of the most intuitive, popular and effective remedies to accelerate RL (Dorigo and Colombetti 1994; Mataric 1994; Randløv and Alstrøm 1998). It also played an important role in recent successes like Doom (Wu and Tian 2017) and Dota (OpenAI et al. 2019).

However, most shaping functions are *hand-designed* by human experts and are potentially tedious and inconvenient to code in especially complex environments (Wu and Tian 2017; OpenAI et al. 2019). Direct shaping functions may also change the optimal policies (Ng, Harada, and Russell 1999). Furthermore, in practice we are usually interested in solving multiple similar tasks as a whole (e.g., training *one* general agent for all possible 2D mazes instead of one per map). Such shared but not identical task-structures naturally induce a *distribution over tasks*, such as distributions over maze configurations (Wilson et al. 2007), system parameters for different robot-hand sizes (Lazaric and Ghavamzadeh 2010) and game maps for RTS games (Jaderberg et al. 2019). The ability to quickly solve new similar tasks drawn from such distributions is mastered by human infants quite young (Smith and Slone 2017). However, the human effort in reward shaping for RL would be further exacerbated as we have to either design a different shaping per task or come up with a general shaping function presumably harder to design.

Motivated by such inconvenience under task multiplicity, we study the generally hard problem of learning automatic reward shaping on a distribution of tasks (termed *learning task-distribution reward shaping*). Our goal is not to propose new general meta-learning algorithms, but instead use them to learn good shaping. To the best of our knowledge, the only previous such works are (Konidaris and Barto 2006; Snel and Whiteson 2014), both using naive objectives and simple models. They first learn optimal Q-/V-values for each task and then fit one potential function to all the learned values. They have to learn as many value functions as tasks in the first step. More importantly, some tasks' value functions might be in disagreement with others' and would be inappropriately averaged out in the second step. Besides, its training doesn't prepare for adaptation to individual tasks at all. To mitigate those problems, our main **novelties** are an overall framework that formulates task-distribution reward shaping as meta-learning, and an original algorithm for reward shaping in meta-testing. Our **contributions** include: 1) we present a state-of-the-art method using meta-learning and deep nets to learn a *general and flexible* task-distribution reward shaping, requiring *only* shared state space and applicable to new tasks either directly or adaptively; 2) we provide insightful theory, and extensive empirical improvements over strong competitors including Rainbow (Hessel et al. 2018), MQL (Fakoor et al. 2020), hand-designed shaping and learning intrinsic rewards (Zheng et al. 2020).

2 Preliminaries

We consider the setting of multi-task model-free RL, where the tasks follow a distribution $p(\mathcal{T})$. Each sampled task

*J.Z is the corresponding author.

$\mathcal{T}_i \sim p(\mathcal{T})$ is a standard Markov Decision Process (MDP) $M_i = (\mathcal{S}, \mathcal{A}_i, T_i, \gamma, R_i)$, where $\mathcal{S}$ is state space, assumed to be shared by all tasks, $\mathcal{A}_i$ is the action space, $T_i : \mathcal{S} \times \mathcal{A}_i \times \mathcal{S} \to [0,1]$ is the unknown state transition probability (hence model-free), $\gamma \in [0,1]$ is the discount factor and $R_i : \mathcal{S} \times \mathcal{A}_i \times \mathcal{S} \to \mathbb{R}$ is the reward function. Note we use the subscript i to denote potentially *different* action spaces $\mathcal{A}_i$, transition probabilities T_i and reward functions R_i. We assume basic knowledge of (optimal) value functions and (deep) Q-learning, and briefly introduce the techniques on which our method is based as follows.

2.1 Potential-based Shaping Function

A shaping function F transforms the original MDP $M = (\mathcal{S}, \mathcal{A}, T, \gamma, R)$ into a **shaped** MDP $M' = (\mathcal{S}, \mathcal{A}, T, \gamma, R' = R + F)$. An arbitrary F could change the original optimal policies. One particular form of F, potential-based shaping function (Ng, Harada, and Russell 1999), keeps the optimal policies invariant (Policy invariance theorem (Ng, Harada, and Russell 1999)):

Definition 1 (Potential-based shaping function (Ng, Harada, and Russell 1999))**.** *$F : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a potential-based shaping function if there exists a real-valued function $\Phi : \mathcal{S} \to \mathbb{R}$, such that $\forall (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$,*

$$F(s, a, s') = \gamma \Phi(s') - \Phi(s).$$

$\Phi(s)$ is thus called the potential function.

Ng, Harada, and Russell (1999) noted $\Phi(s) = V^*_M(s)$ (the optimal state value) gives $V^*_{M'}(s) \equiv 0$ and leaves only the non-zero Q-values to learn. We provide further insightful analysis and motivate it in our method in §3.1.

2.2 Meta-Learning

Meta-learning has proved effective for task distributions in RL (Duan et al. 2016; Wang et al. 2016a; Yu et al. 2019). It operates on two task sets: **meta-training** set $\{\mathcal{T}_i\}_{i=1}^N$ and **meta-testing** set $\{\mathcal{T}_j\}_{j=N+1}^{N+M}$, both drawn from the same task distribution $p(\mathcal{T})$. One popular algorithm is Model-Agnostic Meta-Learning (MAML) (Finn, Abbeel, and Levine 2017):

In meta-training, MAML learns a parameter initialization θ of a neural net f, iterating in "meta-iterations" within which it samples a minibatch of tasks $\{\mathcal{T}_i\}$ and updates:

$$\phi_i(\theta) \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta), \quad (1)$$

$$\theta \leftarrow \theta - \beta \nabla_\theta \mathbb{E}_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_i(\theta)}). \quad (2)$$

Here θ are the parameters to be learned, ϕ_i are the task-specific parameters updated from θ as initialization (Eqn. (1)), α, β are step sizes and $\mathcal{L}_{\mathcal{T}_i}$ is the loss function on each $\mathcal{T}_i$. Note that ϕ_i depend on θ (hence the explicit $\phi_i(\theta)$ notation) and the gradients back-propagate through ϕ_i to θ in Eqn. (2). In other words, this learning procedure is learning θ such that a single fine-tune step $\phi_i(\theta)$ could effectively reduce the loss on a specific task $\mathcal{T}_i$.

In meta-testing, given data from a new task $\mathcal{T}_j$, MAML adapts (with several updates) model parameters starting from θ, learning much faster than from random initialization.

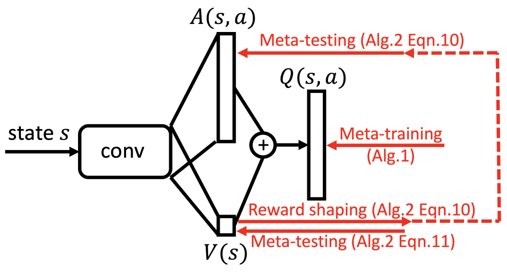

$A(s,a)$
Meta-testing (Alg.2 Eqn.10)
$Q(s,a)$
state s
conv
Meta-training (Alg.1)
Reward shaping (Alg.2 Eqn.10)
Meta-testing (Alg.2 Eqn.11)
$V(s)$

Figure 1: Overview of the proposed framework. The dueling-DQN architecture is shown in black. The forward/backward computation flow is shown in red, where meta-training (Alg. 1) has a simpler flow than meta-testing (Alg. 2).

3 Methods

We propose a novel framework (depicted in Fig. 1) to learn task-distribution reward shaping. It casts task-distribution reward shaping onto meta-learning and learns a potential function $\Phi(s)$ for reward shaping on newly sampled tasks. In essence, we would parameterize $\Phi(s)$ as a dueling-DQN[1] (Wang et al. 2016b) and meta-learn its parameter initialization during meta-training. We then propose a novel algorithm to adapt from the learned initialization for reward shaping during meta-testing.

We generally follow the notations in §2 and summarize the main notations to appear for reference in a table in our arXiv appendix. Since we seek to extract *prior* knowledge from meta-training tasks $\{M_i\}$ into $\Phi(s)$, we call $\Phi(s)$ the **prior** (also following recent works (Yoon et al. 2018; Finn, Xu, and Levine 2018)). Correspondingly, in *meta-testing* we adapt it towards the **task-posterior** $\Phi(s|\mathcal{T}_i)$ for better shaping. We parameterize the prior with θ: $\Phi(s;\theta)$, and the task-posterior with ϕ_i: $\Phi(s|\mathcal{T}_i;\phi_i)$.[2] We use the same letter Φ for both prior and posterior because they would have the same neural network architecture though different parameter values. We use the same parameters θ, ϕ_i and ϕ_j in equations like $Q_{\phi_i}(s,a) = V_{\phi_i}(s) + A_{\phi_i}(s,a)$ for simplicity due to the dueling architecture and shared parameters. These would become clear later.

We first motivate our choice of the task-posterior $\Phi(s|\mathcal{T}_i)$.

3.1 Efficient Credit Assignment with Optimal Potential Functions

A key insight on the potential function $\Phi(s) = V^*_M(s)$ is:

Theorem 1 (Dependency Removal, proof in arXiv appendix)**.** *For all states, $\Phi(s) = V^*_M(s)$ shaping gives $R'(s,a) = \mathbb{E}_{s'} R'(s,a,s') \le 0$ (non-positive immediate rewards) with optimal actions' immediate rewards being exclusively* 0.

It essentially removes *long-term dependencies* along trajectories, since immediate rewards (zero or negative) are

[1]To be more specific, its V-value head would serve as the potential function.

[2]We implement them as ordinary neural networks rather than distributions for optimization simplicity. We leave a fuller Bayesian treatment than such point estimates for future work.

Algorithm 1 Meta-learning potential function prior

```
Input: $p(\mathcal{T})$: a distribution over tasks; $\alpha$, $\beta$: step sizes
Output: Learned prior MD3QN $Q_\theta$
Randomly initialize parameters $\theta$ for prior $Q_\theta$
for meta_iteration $= 0, 1, 2...$ do
    Sample a mini-batch of tasks $\{\mathcal{T}_i\} \sim p(\mathcal{T})$
    for all $\mathcal{T}_i$ do
        Initialize replay buffer $\mathcal{D}_i$
        Collect data with $Q_\theta(s, a)$ ($\epsilon$-greedy) into $\mathcal{D}_i$
        $\phi_i(\theta) \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}^Q_{\mathcal{T}_i}(Q_\theta)$ (Eqn. (8) with $\mathcal{D}_i$)
    end for
    $\theta \leftarrow \theta - \beta \nabla_\theta \mathbb{E}_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(Q_{\phi_i(\theta)})$ (Eqn.(9), all $\{\mathcal{D}_i\}$)
end for
```

sufficient to determine optimal policies. There's then no need of *credit assignment*, hence the arguably optimal credit assignment for learning efficiency.

Therefore, we choose $V^*_{M_i}(s)$ as the adaptation target of the task-posterior $\Phi(s|\mathcal{T}_i)$ to be used in Eqn. (1).

Remarks: 1) This simple result has been noted similarly, though not identical, in e.g. (Baird III 1993; Schulman et al. 2016). Here it mainly introduces our adaptation target. 2) Thm. 1 only holds in expectation of environmental randomness, and has to be approximated by mini-batches of samples. Still, $V^*_{M_i}(s)$ is a much appealing choice. 3) Knowing V^* seems equivalent to having solved the task partly (V^* still cannot give policies in the model-free setting), but we are not using it for the same task in a circle: $V^*_{M_i}(s)$ serves as the learning target on meta-training tasks $\{M_i\}$ in order to better *generalize* and approximate it for shaping on meta-testing tasks $\{M_j\}$.

3.2 Meta-Learning Potential Function Prior

Having set the adaptation update (Eqn. (1)), we now consider how to learn the prior (Eqn. (2)) taking into account the adaptation and reward shaping. We would arrive at Alg. 1.

We design $\Phi(s;\theta)$ and $\Phi(s|\mathcal{T}_i;\phi_i)$ to be of the *same* network architecture (hence both Φ). For each $\mathcal{T}_i$, ϕ_i is initialized to θ and adapts to $V^*_{M_i}(s)$ under some loss function $\mathcal{L}_{\mathcal{T}_i}$:

$$\phi_i(\theta) \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}\big(\Phi(s|\mathcal{T}_i;\theta)\big). \tag{3}$$

Then from Eqn. (2), our objective for the prior is:

$$\min_\theta \mathbb{E}_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}\big(\Phi(s|\mathcal{T}_i;\phi_i(\theta))\big). \tag{4}$$

In contrast, previous works of learning task-distribution reward shaping (Konidaris and Barto 2006; Snel and Whiteson 2014) use $\min_\theta \mathbb{E}_{\mathcal{T}_i} \mathcal{L}^Q_{\mathcal{T}_i}(Q_\theta)$, which is essentially different and we refer to as NAIVE AVERAGING. Regarding meta-learning, this is actually the "pretraining one network on all training tasks" baseline in (Finn, Abbeel, and Levine 2017), where it's already shown to be inferior to MAML.

Choice of $\mathcal{L}_{\mathcal{T}_i}$ and Φ: The closest off-the-shelf RL way to learn $V^*_{M_i}(s)$ is arguably (deep) Q-learning (Watkins and Dayan 1992; Mnih et al. 2015), which learns $Q^*_{M_i}(s,a)$ and gives $V^*_{M_i}(s)$ simply with $V^*_{M_i}(s) = \max_a Q^*_{M_i}(s,a)$.

Algorithm 2 Meta-testing (adaptation with advantage head)

```
Input: $\mathcal{T}_j$: new task to solve; meta-learned MD3QN $Q_\theta$
Output: adapted task-posterior $\phi_j$
Initialize $\phi_j \leftarrow \theta$ and replay buffer $\mathcal{D}$
for gradient_step $= 0, 1, 2...$ do
    Collect data with $A_{\phi_j}(s, a)$ into $\mathcal{D}$
    Update $A_{\phi_j}(s, a)$ with Eqn. (10), using samples from $\mathcal{D}$ and the current potential function $V_{\phi_j}(s)$ for shaping
    Update $V_{\phi_j}(s)$ with Eqn. (11), using samples from $\mathcal{D}$
end for
```

Therefore, we use a deep Q-network (DQN) (Mnih et al. 2015) parameterized as Q_θ and Q_{ϕ_i} for the potential:

$$\Phi(s|\mathcal{T}_i;\phi_i) = \max_a Q_{\phi_i}(s,a) \tag{5}$$

Correspondingly, $\mathcal{L}_{\mathcal{T}_i}$ is the (deep) Q-learning objective on Q instead of Φ:

$$\mathcal{L}^Q_{\mathcal{T}_i}(Q_{\phi_i}) = \mathbb{E}_{\mathcal{D}_i} \|R_i(s,a,s') + \gamma \max_{a'} Q_{\phi_i}(s',a') - Q_{\phi_i}(s,a)\|^2,$$

computed on some sampled (off-policy) trajectory data $\mathcal{D}_i$.

Remark: We could also use modern policy-gradient algorithms (Mnih et al. 2016; Schulman et al. 2017) that learn $V^\pi(s)$ in the meantime of optimizing the policy $\pi(s)$, although they don't *directly* learn $V^*(s)$ and $V^\pi(s) \to V^*(s)$ only when $\pi(s) \to \pi^*(s)$, which may require more data to learn. Therefore, we only consider DQN variants in this paper, and leave policy-gradients (e.g., off-policy actor-critic (Lillicrap et al. 2016)) for future work.

DQN Design: Taking $\max$ (Eqn. (5)) in vanilla DQNs isn't very robust to approximation error and noise, and the learned Q-values are sometimes not necessarily good estimates of true values (Wang et al. 2016b; Tallec, Blier, and Ollivier 2019). So we further introduce a key design of decomposed multi-head architecture (also justified in §5):

$$Q_{\phi_i}(s,a) = V_{\phi_i}(s) + A_{\phi_i}(s,a), \tag{6}$$

where we'd like the advantage head $A_{\phi_i}(s,a)$ to learn the advantage-value function. This explicit decomposition gives the arguably directest way to learn V^*:
Firstly, note an issue of identifiability in Eqn. (6) that an arbitrary bias could be added to V_{ϕ_i} and subtracted from A_{ϕ_i} while keeping Q_{ϕ_i} unchanged. Therefore, the final design is to subtract the maximum of the advantage value from Q_{ϕ_i}:

$$Q_{\phi_i}(s,a) = V_{\phi_i}(s) + A_{\phi_i}(s,a) - \max_{a'} A_{\phi_i}(s,a'). \tag{7}$$

Then as Q_{ϕ_i} attains $Q^*_{M_i}$, by taking $\max_a$ on both sides of Eqn. (7), we get $V_{\phi_i}(s) = \max_a Q^*_{M_i}(s,a) = V^*(s)$. So we can *directly* learn V^* in the meantime of learning Q^*.

This decomposition directly mimics and is justified by the value functions' relation $Q^\pi_{M_i}(s,a) = V^\pi_{M_i}(s) + A^\pi_{M_i}(s,a)$, by definition for *any* policy π and MDP M_i. It was first introduced as dueling DQN (Wang et al. 2016b) but for a different purpose of speeding up training ("generalize learning across actions" (Wang et al. 2016b)). Here we exploit the architecture in learning the optimal V-values V^* as we meta-train at the combined $Q(s,a)$ (Fig. 1).

Now the potential function becomes $\Phi(s|\mathcal{T}_i;\phi_i) = V_{\phi_i}(s)$, i.e., the $V(s)$ head as shown in Fig. 1.

The **final algorithm** is as Alg. 1. Eqn. (3) and (4) become:

$$\phi_i(\theta) \leftarrow \theta - \alpha\nabla_\theta \mathcal{L}^Q_{\mathcal{T}_i}(Q_\theta), \tag{8}$$

$$\min_\theta \mathbb{E}_{\mathcal{T}_i}\mathcal{L}^Q_{\mathcal{T}_i}(Q_{\phi_i(\theta)}). \tag{9}$$

Roughly, Alg. 1 extends MAML and dueling-DQN for our purposes. However, since DQN is *off-policy RL*, we could reuse experience and require much less experience every meta-iteration than the original (on-policy) policy-based MAML (Finn, Abbeel, and Levine 2017), which re-samples data in each environment with the task-posterior ϕ_i for the update in Eqn. (9).

In practice, we also incorporate value-estimation corrections Double DQN (Van Hasselt, Guez, and Silver 2016) and multi-step returns (Hessel et al. 2018), resulting in multi-step Dueling Double DQN (MD3QN). A baseline without dueling is thus MDDQN. It's non-trivial to combine other parts of Rainbow (Hessel et al. 2018) with meta-learning, and meta-learning the simpler MD3QN already outperforms RAINBOW + NAIVE AVERAGING (§5.3).

3.3 Meta-Testing with Potential Function Prior

Naturally we would like to adapt the θ-initialized shaping $V_{\phi_j}(s)|_{\phi_j=\theta}$ towards $V^*_{M_j}(s)$ when learning on a meta-testing task $\mathcal{T}_j$. To facilitate this, we made an additional assumption first that the action space *is* shared across the task distribution (we'll drop it and handle the unshared case later), so the MD3QN is still applicable on $\mathcal{T}_j$. We call this meta-testing setting **adaptation with advantage head**, as later in the unshared case we would discard the advantage head.

We'd like an algorithm that 1) utilizes the meta-trained MD3QN and 2) still does RL on the shaped M'_j and 3) adapts the shaping function $V_{\phi_j}(s)$ in a simultaneous but also clear, disentangled manner. We propose Alg. 2, which updates A_{ϕ_j}, V_{ϕ_j} alternately with stepsize α:

○ Update $A_{\phi_j}(s,a)$ with sampled data from replay buffer:

$$\phi_j \leftarrow \phi_j - \alpha\nabla_{\phi_j}\mathbb{E}_{\mathcal{D}}\|R'_j(s,a,s') + \gamma\max_{a'}A_{\phi_j}(s',a') - A_{\phi_j}(s,a)\|^2. \tag{10}$$

○ Update $V_{\phi_j}(s)$ towards target y with sampled data from replay buffer (where $y = \max_a A_{\phi_j}(s,a) + V_{\phi_j}(s)$):

$$\phi_j \leftarrow \phi_j - \alpha\nabla_{\phi_j}\mathbb{E}_{\mathcal{D}}\|V_{\phi_j}(s) - y\|^2. \tag{11}$$

They update intermediate $A(s,a)$ and $V(s)$ heads (Fig. 1).

Theorem 2 (Meta-Testing Objective, proof in arXiv appendix)**.** *Eqn.*(10) *optimizes for the optimal policy. Eqn.*(11) *optimizes for the task-posterior* $V^*_{M_j}(s)$.

We now return to unshared action spaces, which could arise in practice simply because the task requires a different action space (e.g., from discrete to fine-grained continuous spaces in §5.5), or because of other constraints forbidding the advantage head. In this case (which we call **shaping only**), we simply discard the advantage head and fix the meta-learned V_θ for reward shaping to train a new RL agent from scratch on $\mathcal{T}_j$. Due to the close (though nonequivalent) relation between Alg. 1 and Q-learning, V_θ could still be expected to be much better than random potential functions or no shaping at all, which we empirically verify in our arXiv appendix.

Meta-testing without reward shaping: In case of shared action space, one may wonder: why not simply do MAML's meta-testing (Finn, Abbeel, and Levine 2017), just Q-learning starting from θ? We refer to this baseline as VANILLA MAML. Though viable, VANILLA MAML merely exploits the meta-learned parameter initialization, while our Alg. 2 *also* explicitly exploits reward shaping from the potential function prior. The shaped rewards are easier for policy learning, and the adapting shaping (Eqn. (11)) further boosts policy learning (Eqn. (10)) immediately in the next loop. It also jointly does RL and adapts V_{ϕ_j} in a much more clear, disentangled manner. These are empirically supported by the faster and more stable curves of our method in §5.3. Also, note that we are more general on action spaces, while VANILLA MAML requires a shared one.

4 Related Work[3]

Learning task-distribution reward shaping: The only direct previous works are (Konidaris and Barto 2006; Snel and Whiteson 2014). They do consider multi-task transfer, but are based on naive (weighted) averaging in principle (whose downsides are discussed in §1) and only learn simple linear models on simple tasks. We include them in §5 as all NAIVE AVERAGING baselines with our deep models.

(Automatic) reward shaping: Potential-based reward shaping with policy invariance guarantee (Ng, Harada, and Russell 1999) have been extended to various settings (Asmuth, Littman, and Zinkov 2008; Lu, Schwartz, and Givigi 2011; Gao and Toni 2015; Eck et al. 2016; Mannion, Duggan, and Howley 2017; Grześ 2017) and applications (Devlin, Grześ, and Kudenko 2011; Efthymiadis and Kudenko 2013). The exact form of the potential function could also be generalized to depend on action and/or timestep (Wiewiora, Cottrell, and Elkan 2003; Devlin and Kudenko 2012; Harutyunyan et al. 2015) in addition to state. Here we simply stick with its original form in (Ng, Harada, and Russell 1999). Potential-based shaping functions are usually pre-defined manually. Earlier attempts of automatic reward shaping rely on pre-defined representations (Konidaris and Barto 2006; Marthi 2007), pre-defined higher-level shaping function (Laud and DeJong 2002) or evolutionary search of shaping functions (Niekum, Barto, and Spector 2010). Above all, those works are almost entirely based on tabular RL, lacking the ability to generalize/adapt. One recent related work on a task distribution is (Jaderberg et al. 2019), but they still learn in a tabular way on a limited set of predefined novel states of their task.

In contrast to all those works, our method is quite general, assuming no task knowledge or heuristics, with a more general, principled meta-learning objective, flexible application settings, insightful theoretical analysis, and both neural agents and shaping functions with gradient optimization.

[3]We include some additional related work on reward shaping, meta-learning and inverse RL in our arXiv appendix.

Another line of work is **learning intrinsic rewards** (Barto 2013), which learns the reward itself for policy optimization. It *replaces* the original reward and is fundamentally different from reward shaping. Policy invariance are usually only guaranteed asymptotically. Some learn on the program/code level (Alet* et al. 2020). Some are for single tasks (Zheng, Oh, and Singh 2018; Stadie, Zhang, and Ba 2020; Du et al. 2019). Works comparable with ours on task distributions (Yang et al. 2019; Zheng et al. 2020) build on on-policy RL (§5.3).

5 Experiments

This sections seeks to answer two key questions: 1) what exactly is being learned and 2) how our framework compares to relevant baselines. We experimented on (of increasing complexity) CartPoles, grid mazes and CoinRun games through comparison with the only previous baselines (Konidaris and Barto 2006; Snel and Whiteson 2014) and other strong competitors (incl. RAINBOW (Hessel et al. 2018), hand-designed shaping, MQL (Fakoor et al. 2020) and intrinsic rewards (Zheng et al. 2020)). As a reminder, our method ("OURS") meta-trains a MD3QN with Alg. 1, and meta-tests with Alg. 2 in the **adaptation with advantage head** (in short, **adaptation**) case or with the meta-learned V_θ in the **shaping only** case. We've included further experimental details and results in our arXiv appendix.

5.1 General Settings

We obtain each result over 5 different runs (random seeds 0-4). We sample, separate and fix the meta-training and meta-testing tasks at the beginning of each run, keeping all hyperparameters the same. For fair comparison, meta-training of all methods uses *same number* of environment interactions and updates. For each meta-testing curve (# gradient steps as the x-axis in e.g., Fig. 3, 4), we keep all data collection hyperparameters the same across all methods (e.g., # environment interactions per update) so that the number of gradient steps accurately mirrors *sample complexity*. For CartPoles and grid mazes, we aggregate raw performance statistics on all meta-testing tasks combined (so e.g., $5 \times 40 = 200$ tasks on CartPoles as follows), and then take median and interquartile curves. For CoinRun we take mean and standard deviation of all runs' solve ratios following (Cobbe et al. 2019).[4]

The task distribution on CartPoles is defined varying the pole length in the range of $[0.25, 5.00]$. Such change in dynamics is already harder than random noise. The mD3QN uses an MLP with two hidden layers of size 32 before the value heads. We meta-train on 500 sampled tasks for 200 meta iterations with 10 tasks per iteration and meta-test on 40 unseen tasks.

The task distribution on grid mazes is defined on all possible maps of size 8×8, which is of exponentially many configurations varying start, goal and obstacles. The mD3QN uses a CNN similar as (Mnih et al. 2015) before the value heads. We meta-train on 1000 sampled maps for 1000 meta iterations and meta-test on 40 unseen maps. Pixel grid mazes is already non-trivial for non-tabular agents (Tamar et al. 2016).

[4]Mean + std curves for CartPoles and grid mazes are also included in our arXiv appendix, giving the same conclusions. It also shows we're robust across different evaluation metrics.

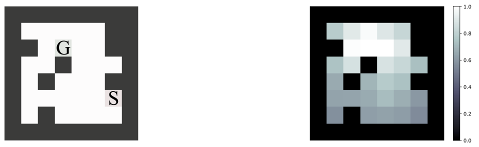


Figure 2: Visualization of meta-learned prior V_θ (right) on the left map at all possible agent positions.

The task distribution on CoinRun games (Cobbe et al. 2019) is defined on all possible level configurations. It's a very difficult pixel game (Cobbe et al. 2019). The mD3QN uses pixel input and is also a CNN. We meta-train on 2000 generated levels for 1500 meta iterations. Meta-testing is done on a held-out set of 1000 levels.

Computationally, meta-learning is ˜20% slower than non-meta-learning baselines due to the two-level MAML update, but on par with the more complex Rainbow (Hessel et al. 2018). Meta-testing is similar for all methods, only a bit slower than no shaping.

5.2 Qualitative Investigation of Learned Shaping

A natural question is **what exactly is being learned** in meta-training. We therefore plot the meta-learned prior, V_θ, on one randomly sampled unseen grid maze (from start (S) to goal (G)) in Fig. 2 by iterating the agent location over all viable grid positions to get the value prediction for each state. We can see that the prior learns to generalize over state representations: plotted values overall match true values under the setting of +1 reward at goal, 0 reward elsewhere and discount factor 0.99. Although the map configuration varies across the task distribution, the representations of the start, goal, obstacles and agent positions are the same and meta-training could generalize over the detailed representations.

The prior (Fig. 2) is good for **shaping only**, but are not perfect (so could benefit from **adaptation**): they discount more rapidly than 0.99, and not all positions with same distance to goal have same values, especially farther away from goal. These match our analysis in §3.3. We also observe that some baselines we consider could get similar visualizations but with poorer quality.

On CoinRun we observe that the learned shaping is more complex since the game requires more maneuver, but it generally tends to encourage agent movement towards goal.

Another natural question is: is our learned shaping better than **pure random shaping**, since optimal policies are preserved under arbitrary potential-based reward shaping? we ablate with two random baselines to prove the effectiveness of our learned shaping:

○ RANDOM NOISE SHAPING: generate random noise and substitute the reward shaping whenever needed.

○ RANDOM NETWORK SHAPING: reward shaping with a randomly initialized MD3QN directly used.

We experiment on grid mazes. Expectedly, the random baselines learn very poorly in both the **adaptation** and **shaping only** cases. The median returns remain zero until the

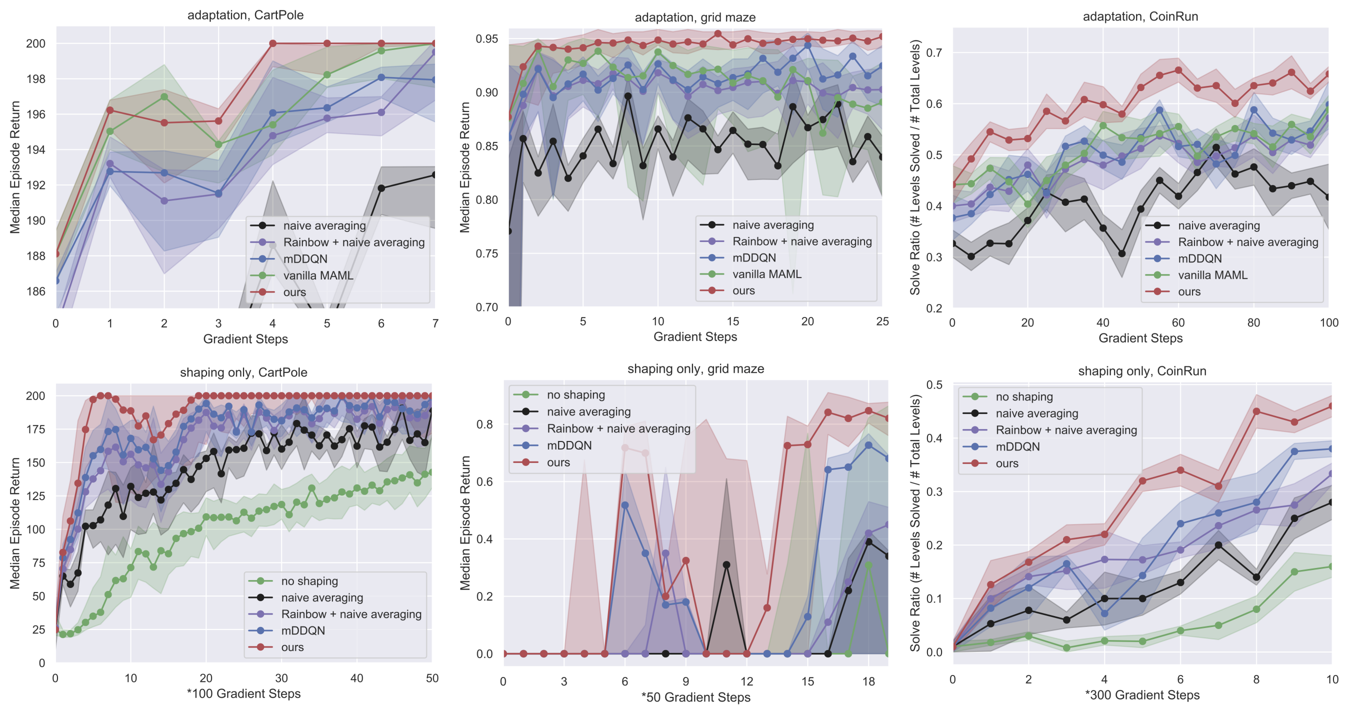


Figure 3: Meta-testing learning curves on distributions of (by column) CartPoles, grid mazes and CoinRun games. Top row: adaptation case (POLICY-BASED MAML performs too badly to appear in range due to its on-policy nature); Bottom row: shaping only case. OURS (red) achieves state-of-the-art reward shaping and outperforms all baselines.

same update number as of Fig. 3 (grid maze column). This proves that our learned shaping which generalizes over state representations does provide meaningful rewards.

5.3 Meta-Testing Performance

The key evaluation is learning curve comparison in both cases of meta-testing. We consider mostly the baselines emerging from §3 with their difference from OURS as follows:

○ NAIVE AVERAGING (Konidaris and Barto 2006; Snel and Whiteson 2014): its prior-learning objective is $\min_\theta \mathbb{E}_{\mathcal{T}_i} \mathcal{L}^Q_{\mathcal{T}_i}(Q_\theta)$ instead of Eqn. (9) as discussed in §3.2. Its original version needs to first learn the same number of value functions as the number of tasks, which does not scale well to large number of tasks. We instead merge its two steps and directly train an RL agent in all meta-training tasks, which is theoretically equivalent. This is realized by simply using only one replay buffer for all tasks and do conventional deep Q-learning with it. (Cobbe et al. 2019) studies the same approach on CoinRun but with different setting details.

○ RAINBOW + NAIVE AVERAGING: same as NAIVE AVERAGING but with better agent design (Hessel et al. 2018).

○ MDDQN: taking away only the dueling part; to be compatible with Alg. 2 we further introduce $V_{\phi_j}(s) = \max_a Q_{\phi_j}(s,a)$ and $A_{\phi_j}(s,a) = Q_{\phi_j}(s,a) - V_{\phi_j}(s)$.

○ VANILLA MAML: (only in **adaptation** case) adapts without shaping as (Finn, Abbeel, and Levine 2017).

○ POLICY-BASED MAML: (only in **adaptation** case) the original MAML-RL (Finn, Abbeel, and Levine 2017), which is policy-based instead of value-based.

○ NORML: (only in **adaptation** case) as (Yang et al. 2019), does the inner update of POLICY-BASED MAML with a learnable advantage function without external rewards.

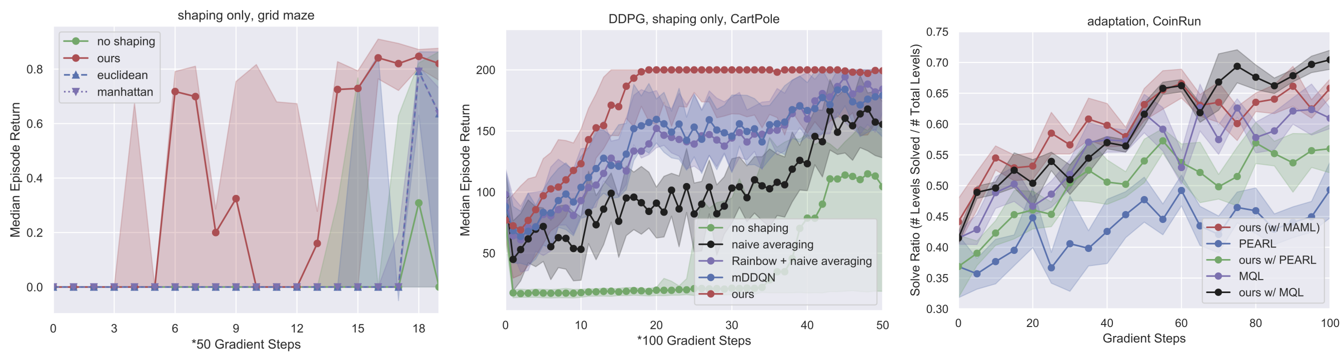
○ LIRPG-META: (only in **shaping only** case) as (Zheng et al. 2020), meta-learns an intrinsic reward function that replaces external rewards (more details in our arXiv appendix).

Remark: The last three baselines (from POLICY-BASED MAML) are all essentially *on-policy*. They uses REINFORCE (Williams 1992) / TRPO (Schulman et al. 2015) / PPO (Schulman et al. 2017) for Eqn. (1) and Eqn. (2). In our experiments, they require more times the amount of training to achieve comparable performance. For clarity we exclude them from the main plots and leave them to our arXiv appendix.

We meta-train all methods following §5.1 and plot meta-testing performances in Fig. 3. In **shaping only** case, training identically randomly-initialized RL agents, we also plot training without any reward shaping ("NO SHAPING"). Note that oscillation could not be completely avoided since it's to some extent inherent to off-policy RL algorithms (Q-learning here).

Across all task distributions and both cases, OURS achieves state-of-the-art in learning task-distribution reward shaping, outperforming previous such works (Konidaris and Barto 2006; Snel and Whiteson 2014). Comparison with MDDQN and VANILLA MAML justifies our dueling design and Alg. 2 reward shaping respectively. We reiterate that pixel-based grid mazes is already non-trivial for non-tabular agents (Tamar et al. 2016), let alone CoinRun.

Note that in our setting the amount of meta-training expe-



(a) §5.4 Hand-designed Shaping (b) §5.5 DQN to DDPG (c) §5.6 PEARL & MQL

Figure 4: Additional meta-testing learning curve comparisons.

rience and update steps per task on average are quite limited compared to conventional deep RL. This may be another reason why NAIVE AVERAGING performs poorly, and RAINBOW + NAIVE AVERAGING improves it only to some extent. The last three *on-policy* competitors also requires more training. They do catch up when we train them for much longer with more data and updates.

5.4 Comparison with Hand-designed Shaping

We further experimented on the well-studied domain for reward shaping, grid mazes. Manhattan or Euclidean distance to goal is commonly used as the potential function (Ng, Harada, and Russell 1999; Grzes and Kudenko 2009; Asmuth, Littman, and Zinkov 2008; Devlin and Kudenko 2012; Marom and Rosman 2018). Since the distance functions cannot adapt, we only compare in **shaping only** case. From Fig. 4a, both distances are better than NO SHAPING, but OURS performs even better, which is quite significant.

Bit surprisingly, the two distances perform almost identically, possibly because they are somewhat similar (L1/L2 distance) for reward shaping in 2D mazes. In this paper we always train generalizable neural agents. Here tabular agents would of course learn very fast, but they cannot generalize, use specific tabular structure and are not a fair comparison.

5.5 Different Action Spaces in Meta-Testing

We also test under unshared action space. We model this by meta-testing in a continuous action space on CartPoles. We let it take as input actions of continuous values. This is also the **shaping only** case. After meta-training, we train DDPG agents from scratch on meta-testing tasks. Clearly from Fig. 4b, our reward shaping lets the agent achieve max return the fastest. The initial policy appears better than in the discrete case because we apply `tanh` to bound the action output. However, due to the harder rewards to learn, other methods struggle in early stages.

5.6 PEARL and MQL on CoinRun

In the **adaptation** case, we further consider PEARL (Rakelly et al. 2019) and MQL (Fakoor et al. 2020) that are shown to outperform MAML on some but not all tasks (Yu et al. 2019). Generally, they both use the NAIVE AVERAGING (multi-task) objective and condition the RL agent on episode history so the problems of the naive objective are mitigated. The following comparison and combination with OURS are considered:

∘ PEARL, MQL: their meta-training/testing algorithms but substituting their RL part with MD3QN.

∘ OURS + PEARL, OURS + MQL: our Alg. 1 & 2 but substituting our meta-learning part with PEARL / MQL. Meta-training is the same as the above PEARL / MQL respectively (but not meta-testing, more details in our arXiv appendix).

We again meta-train all methods fairly following §5.1 and show meta-testing results in Fig. 4c. We outperform almost all the methods. On one hand, we surpass PEARL and MQL, though MQL is indeed competitive. Since generally our meta-learning part, MAML, is at lesat not much better than PEARL and MQL, our better result could only be attributed mainly to the reward shaping part. On the other hand, both PEARL and MQL are boosted when combined with OURS, with OURS + MQL finally better than OURS. This again justifies our reward shaping. Again we note that our goal is not to propose a new general meta-learning algorithm as PEARL / MQL, but to learn task-distribution reward shaping with meta-learning.

6 Discussions

Our method builds on meta-learning, and a common question about meta-learning is on what task distributions meta-learning could work (Yu et al. 2019). Although we build on MAML, our experiments already show we could generalize on task distributions more difficult than most simple distributions varying only reward functions in e.g. (Finn, Abbeel, and Levine 2017; Rakelly et al. 2019): in our case the underlying transition/dynamics vary with different CartPole lengths and the grid and CoinRun configurations, which is essentially more diverse than simply varying target velocities/goal locations in one same environment (Finn, Abbeel, and Levine 2017; Rakelly et al. 2019). However, as empirically tested (Yu et al. 2019), being built on MAML still suggests we couldn't generalize well to ultimately *essentially different* tasks ("non-parametric variation" (Yu et al. 2019)), e.g. all robot-hand pushing, sliding and pressing tasks combined.

Acknowledgements

This work was supported by NSFC Projects (Nos. 62061136001, 61620106010, U19B2034, U1811461), National Key Research and Development Program of China under grant 2020AAA0106000, Beijing NSF Project (No. L172037), Beijing Academy of Artificial Intelligence (BAAI), and the Xplorer Prize.

Haosheng Zou would personally like to thank his wife, Jiamin Deng, for her incredible support during the whole process of this paper.

References

Alet*, F.; Schneider*, M. F.; Lozano-Perez, T.; and Kaelbling, L. P. 2020. Meta-learning curiosity algorithms. In *ICLR*.

Asmuth, J.; Littman, M. L.; and Zinkov, R. 2008. Potential-based Shaping in Model-based Reinforcement Learning. In *AAAI*.

Baird III, L. C. 1993. Advantage updating. Technical report, WRIGHT LAB WRIGHT-PATTERSON AFB OH.

Barto, A. G. 2013. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, 17–47. Springer.

Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; and Schulman, J. 2019. Quantifying Generalization in Reinforcement Learning. In *ICML*.

Devlin, S.; Grześ, M.; and Kudenko, D. 2011. Multi-agent reward shaping for robocup keepaway. In *AAMAS*.

Devlin, S.; and Kudenko, D. 2012. Dynamic potential-based reward shaping. In *AAMAS*.

Dorigo, M.; and Colombetti, M. 1994. Robot shaping: Developing autonomous agents through learning. *Artificial intelligence* 71(2): 321–370.

Du, Y.; Han, L.; Fang, M.; Liu, J.; Dai, T.; and Tao, D. 2019. LIIR: Learning Individual Intrinsic Reward in Multi-Agent Reinforcement Learning. In *NeurIPS*.

Duan, Y.; Schulman, J.; Chen, X.; Bartlett, P. L.; Sutskever, I.; and Abbeel, P. 2016. RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv preprint arXiv:1611.02779* .

Eck, A.; Soh, L.-K.; Devlin, S.; and Kudenko, D. 2016. Potential-based reward shaping for finite horizon online pomdp planning. In *AAMAS*.

Efthymiadis, K.; and Kudenko, D. 2013. Using plan-based reward shaping to learn strategies in starcraft: Broodwar. In *2013 IEEE Conference on Computational Intelligence in Games (CIG)*.

Fakoor, R.; Chaudhari, P.; Soatto, S.; and Smola, A. J. 2020. Meta-Q-Learning. In *ICLR*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*.

Finn, C.; Xu, K.; and Levine, S. 2018. Probabilistic Model-Agnostic Meta-Learning. In *NeurIPS*.

Gao, Y.; and Toni, F. 2015. Potential based reward shaping for hierarchical reinforcement learning. In *IJCAI*.

Grześ, M. 2017. Reward shaping in episodic reinforcement learning. In *AAMAS*.

Grzes, M.; and Kudenko, D. 2009. Theoretical and empirical analysis of reward shaping in reinforcement learning. In *International Conference on Machine Learning and Applications*.

Harutyunyan, A.; Devlin, S.; Vrancx, P.; and Nowé, A. 2015. Expressing arbitrary reward functions as potential-based advice. In *AAAI*.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*.

Jaderberg, M.; Czarnecki, W.; Dunning, I.; Marris, L.; Lever, G.; Castañeda, A. G.; Beattie, C.; Rabinowitz, N. C.; Morcos, A. S.; Ruderman, A.; Sonnerat, N.; Green, T.; Deason, L.; Leibo, J. Z.; Silver, D.; Hassabis, D.; Kavukcuoglu, K.; and Graepel, T. 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* 364: 859–865.

Konidaris, G.; and Barto, A. 2006. Autonomous shaping: Knowledge transfer in reinforcement learning. In *ICML*.

Laud, A.; and DeJong, G. 2002. Reinforcement learning and shaping: Encouraging intended behaviors. In *ICML*.

Lazaric, A.; and Ghavamzadeh, M. 2010. Bayesian multi-task reinforcement learning. In *ICML*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *ICLR*.

Lu, X.; Schwartz, H. M.; and Givigi, S. N. 2011. Policy invariance under reward transformations for general-sum stochastic games. *Journal of Artificial Intelligence Research* 41: 397–406.

Mannion, P.; Duggan, J.; and Howley, E. 2017. A theoretical and empirical analysis of reward transformations in multi-objective stochastic games. In *AAMAS*.

Marom, O.; and Rosman, B. 2018. Belief reward shaping in reinforcement learning. In *AAAI*.

Marthi, B. 2007. Automatic shaping and decomposition of reward functions. In *ICML*.

Mataric, M. J. 1994. Reward functions for accelerated learning. In *Machine Learning Proceedings 1994*, 181–189. Elsevier.

Minsky, M. 1961. Steps toward artificial intelligence. *Proceedings of the IRE* 49(1): 8–30.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *ICML*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*.

Niekum, S.; Barto, A. G.; and Spector, L. 2010. Genetic programming for reward function search. *IEEE Transactions on Autonomous Mental Development* 2(2): 83–90.

OpenAI; Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; Józefowicz, R.; Gray, S.; Olsson, C.; Pachocki, J.; Petrov, M.; de Oliveira Pinto, H. P.; Raiman, J.; Salimans, T.; Schlatter, J.; Schneider, J.; Sidor, S.; Sutskever, I.; Tang, J.; Wolski, F.; and Zhang, S. 2019. Dota 2 with Large Scale Deep Reinforcement Learning URL https://arxiv.org/abs/1912.06680.

Rakelly, K.; Zhou, A.; Finn, C.; Levine, S.; and Quillen, D. 2019. Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables. In *ICML*.

Randløv, J.; and Alstrøm, P. 1998. Learning to Drive a Bicycle Using Reinforcement Learning and Shaping. In *ICML*.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *ICML*.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *ICLR*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362(6419): 1140–1144.

Smith, L. B.; and Slone, L. K. 2017. A developmental approach to machine learning? *Frontiers in psychology* 8(2124).

Snel, M.; and Whiteson, S. 2014. Learning potential functions and their representations for multi-task reinforcement learning. In *AAMAS*.

Stadie, B.; Zhang, L.; and Ba, J. 2020. Learning Intrinsic Rewards as a Bi-Level Optimization Problem. In *UAI*.

Tallec, C.; Blier, L.; and Ollivier, Y. 2019. Making Deep Q-learning methods robust to time discretization. In *ICML*.

Tamar, A.; Wu, Y.; Thomas, G.; Levine, S.; and Abbeel, P. 2016. Value iteration networks. In *NeurIPS*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *AAAI*.

Wang, J. X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J. Z.; Munos, R.; Blundell, C.; Kumaran, D.; and Botvinick, M. 2016a. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* .

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016b. Dueling Network Architectures for Deep Reinforcement Learning. In *ICML*.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4): 279–292.

Wiewiora, E.; Cottrell, G. W.; and Elkan, C. 2003. Principled methods for advising reinforcement learning agents. In *ICML*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4): 229–256.

Wilson, A.; Fern, A.; Ray, S.; and Tadepalli, P. 2007. Multi-task reinforcement learning: a hierarchical Bayesian approach. In *ICML*.

Wu, Y.; and Tian, Y. 2017. Training agent for first-person shooter game with actor-critic curriculum learning. In *ICLR*.

Yang, Y.; Caluwaerts, K.; Iscen, A.; Tan, J.; and Finn, C. 2019. NoRML: No-Reward Meta Learning. In *AAMAS*.

Yoon, J.; Kim, T.; Dia, O.; Kim, S.; Bengio, Y.; and Ahn, S. 2018. Bayesian Model-Agnostic Meta-Learning. In *NeurIPS*.

Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; and Levine, S. 2019. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*.

Zheng, Z.; Oh, J.; Hessel, M.; Xu, Z.; Kroiss, M.; van Hasselt, H.; Silver, D.; and Singh, S. 2020. What Can Learned Intrinsic Rewards Capture? In *ICML*.

Zheng, Z.; Oh, J.; and Singh, S. 2018. On learning intrinsic rewards for policy gradient methods. In *NeurIPS*.