Hypernetworks for Zero-Shot Transfer in Reinforcement Learning

Sahand Rezaei-Shoshtari[1,2,3], **Charlotte Morissette**[1,3], **Francois R. Hogan**[3]
Gregory Dudek[1,2,3], **David Meger**[1,2,3]
[1]McGill University
[2]Mila - Québec AI Institute
[3]Samsung AI Center Montreal
srezaei@cim.mcgill.ca

Abstract

In this paper, hypernetworks are trained to generate behaviors across a range of unseen task conditions, via a novel TD-based training objective and data from a set of near-optimal RL solutions for training tasks. This work relates to meta RL, contextual RL, and transfer learning, with a particular focus on zero-shot performance at test time, enabled by knowledge of the task parameters (also known as context). Our technical approach is based upon viewing each RL algorithm as a mapping from the MDP specifics to the near-optimal value function and policy and seek to approximate it with a hypernetwork that can generate near-optimal value functions and policies, given the parameters of the MDP. We show that, under certain conditions, this mapping can be considered as a supervised learning problem. We empirically evaluate the effectiveness of our method for zero-shot transfer to new reward and transition dynamics on a series of continuous control tasks from DeepMind Control Suite. Our method demonstrates significant improvements over baselines from multi-task and meta RL approaches.

1 Introduction

Adult humans possess an astonishing ability to adapt their behavior to new situations. Well beyond simple tuning, we can adopt entirely novel ways of moving our bodies, for example walking on crutches with little to no training after an injury. The learning process that generalizes across all past experience and modes of behavior to rapidly output the needed behavior policy for a new situation is a hallmark of our intelligence.

This paper proposes a strong zero-shot behavior generalization approach based on hypernetworks (Ha, Dai, and Le 2016), a recently proposed architecture allowing a deep hyper-learner to output all parameters of a target neural network, as depicted in Figure 1. In our case, we train on the full solutions of numerous RL problems in a family of MDPs, where either reward or dynamics (often both) can change between task instances. The trained policies, value functions and rolled-out optimal behavior of each source task is the training information from which we can learn to generalize. Our hypernetworks output the parameters of a fully-formed and highly performing policy without any experience in a related but unseen task, simply by conditioning on provided task parameters.

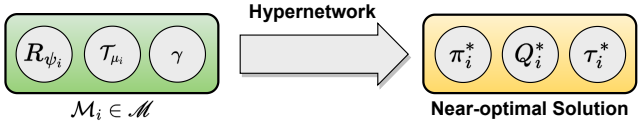


Figure 1: Our method uses hypernetworks to approximate an RL algorithm as a mapping from a family of parameterized MDPs to a family of near-optimal solutions, in order to achieve zero-shot transfer to new reward and dynamics settings.

The differences between the tasks we consider leads to large and complicated changes in the optimal policy and induced optimal trajectory distribution. Learning to predict new policies from this data requires powerful learners guided by helpful loss functions. We show that the abstraction and modularity properties afforded by hypernetworks allow them to approximate RL generated solutions by mapping a parameterized MDP family to a set of optimal solutions. We show that this framework enables achieving strong zero-shot transfer to new reward and dynamics settings by exploiting commonalities in the MDP structure.

We perform experimental validation using several families of continuous control environments where we have parameterized the physical dynamics, the task reward, or both to evaluate learners. We carry out contextual zero-shot evaluation, where the learner is provided the parameters of the test task, but is not given any training time – rather the very first policy execution at test time is used to measure performance. Our method outperforms selected well-known baselines, in many cases recovering nearly full performance without a single timestep of training data on the target tasks. Ablations show that hypernetworks are a critical element in achieving strong generalization and that a structured TD-like loss is additionally helpful in training these networks.

Our main contributions are:

1. The use of hypernetworks as a scalable and practical approach for approximating RL algorithms as a mapping from a family of parameterized MDPs to a family of near-optimal policies.

2. A TD-based loss for regularization of the generated policies and value functions to be consistent with respect to the Bellman equation.
3. A series of modular and customizable continuous control environments for transfer learning across different reward and dynamics parameters.

Our learning code, generated datasets, and custom continuous control environments, which are built upon DeepMind Control Suite, are publicly available at: `https://sites.google.com/view/hyperzero-rl`

2 Background

2.1 Markov Decision Processes

We consider the standard MDP that is defined by a 5-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \text{Dist}(\mathcal{S})$ is the transition dynamics, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and $\gamma \in (0, 1]$ is the discount factor. The goal of an RL algorithm is to find a policy $\pi : \mathcal{S} \to \text{Dist}(\mathcal{A})$ that maximizes the *expected return* defined as $\mathbb{E}_\pi[R_t] = \mathbb{E}[\sum_{k=0}^{T} \gamma^k r_{t+k+1}]$. *Value function* $V^\pi(s)$ denotes the expected return from s under policy π, and similarly *action-value function* $Q^\pi(s, a)$ denotes the expected return from s after taking action a under policy π:

$$Q^\pi(s, a) = \mathbb{E}_{s', r \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')} \Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \Big| s, a \Big]$$

Value functions are fixed points of the Bellman equation (Bellman 1966), or equivalently the Bellman operator $\mathcal{B}^\pi$:

$$\mathcal{B}^\pi[Q(s, a)] = \mathbb{E}_{s', r \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')} \Big[r + \gamma Q(s', a') \Big]$$

Similarly, the optimal value functions $V^*(s)$ and $Q^*(s, a)$ are the fixed points of the Bellman optimality operator $\mathcal{B}^*$.

2.2 General Value Functions

General value functions (GVF) extend the standard definition of value functions $Q^\pi(s, a)$ to entail the reward function, transition dynamics and discount factor in addition to the policy, that is $Q^{\pi,R,\mathcal{T},\gamma}(s, a)$ (Sutton and Barto 2018). Universal value function approximators (UVFA) (Schaul et al. 2015) are an instance of GVFs in which the value function is generalized across goals g and is represented as $Q^\pi(s, a, g)$. Naturally, this notion is used in goal-conditioned RL (Andrychowicz et al. 2017) and multi-task RL (Teh et al. 2017). Relatedly, general policy improvement (GPI) aims to improve a generalized policy based on transitions of several MDPs (Barreto et al. 2020; Harb et al. 2020; Faccio et al. 2022b). The goal of our method in learning a generalized mapping from MDP specifics to near-optimal policies and value functions is closely related to the overall goal of GVFs and GPI. However, unlike such methods we do not seek to improve a given generalized policy.

2.3 Hypernetworks

A *hypernetwork* (Ha, Dai, and Le 2016) is a neural network that generates the weights of another network, often referred to as the *main* network. While both networks have associated weights, only the hypernetwork weights involve learnable parameters that are updated during training. During inference, only the main network is used by mapping an input to a desired target, using the weights generated by the hypernetwork. Since the weights of different layers of the main network are generated through a shared learned embedding, hypernetworks can be viewed as a relaxed form of weight sharing across layers. It has been empirically shown that this approach allows for a level of abstraction and modularity of the learning problem (Galanti and Wolf 2020; Ha, Dai, and Le 2016) which in turn results in a more efficient learning. Notably, hypernetworks can be conditioned on the context vector for conditional generation of the weights of the main network (von Oswald et al. 2019). Similarly to von Oswald et al. (2019), we condition the hypernetwork on the parameters (context) of the MDP to generate the near-optimal policy and value function based on the reward and dynamics parameters.

3 HyperZero

The overarching goal of this work is to develop a framework that allows for approximating RL solutions by learning the mapping between the MDP specifics and the near-optimal policy. A reasonable approximation can potentially allow for zero-shot transfer and predicting the general behaviour of an RL agent prior to its training. Beyond the standard premises of zero-shot transfer learning (Taylor and Stone 2009; Tan et al. 2018), a well-approximated mapping of an MDP to near-optimal policies can have applications in reward shaping, task visualization, and environment design.

3.1 Problem Formulation

This section outlines the assumptions and problem formulation used in this paper. First, we define the *parameterized MDP family* $\mathscr{M}$ as:

Definition 1 (Parameterized MDP Family)**.** A *parameterized MDP family* $\mathscr{M}$ is a set of MDPs that share the same state space $\mathcal{S}$, action space $\mathcal{A}$, a parameterized transition dynamics $\mathcal{T}_\mu$, a parameterized reward function R_ψ, and a discount factor γ:

$$\mathscr{M} = \{\mathcal{M}_i | \mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{T}_{\mu_i}, R_{\psi_i}, \gamma)\},$$

where $\psi_i \sim p(\psi)$ and $\mu_i \sim p(\mu)$ are parameters of $\mathcal{M}_i$, and are assumed to be sampled from prior distributions.

Notably, the state space $\mathcal{S}$ and action space $\mathcal{A}$ in our definition can be either discrete or continuous (e.g., an open sub-space of $\mathbb{R}^n$) spaces. Our definition of a parameterized MDP family is related to contextual MDPs (Hallak, Di Castro, and Mannor 2015; Jiang et al. 2017), where the learner has access to the context.

The key to our approximation is to assume that an RL algorithm, once converged, is a mapping from an MDP $\mathcal{M}_i \in \mathscr{M}$ to a near-optimal policy and the near-optimal action-value function corresponding to the specific MDP $\mathcal{M}_i$ on which it was trained. With a slight abuse of notation, we denote the near-optimal policy as π_i^* and the near-optimal

action-value function as Q_i^*:

$$\mathcal{M}_i \xrightarrow{\text{RL Algorithm}} \pi_i^*(a|s), Q_i^*(s,a). \tag{1}$$

Notably, this view has precedent in prior works on learning to shape rewards (Sorg, Lewis, and Singh 2010; Zheng, Oh, and Singh 2018; Zheng et al. 2020), meta-gradients in RL (Xu, van Hasselt, and Silver 2018; Xu et al. 2020), and the operator view of RL algorithms (Tang, Feng, and Liu 2022).

Since our goal is to learn the mapping of Equation (1) from a family of parameterized MDPs that share the similar functional form of parameterized transition dynamics $\mathcal{T}_\mu$ and reward function R_ψ, we assume that the MDP $\mathcal{M}_i$ can be fully characterized by its parameters ψ_i and μ_i and the functional forms of R_ψ and $\mathcal{T}_\mu$; that is $\mathcal{M}_i \equiv \mathcal{M}(\psi_i, \mu_i)$. Equation (1) can then be simplified as:

$$\mathcal{M}(\psi_i, \mu_i) \xrightarrow{\text{RL Algorithm}} \pi^*(a|s, \psi_i, \mu_i), Q^*(s, a|\psi_i, \mu_i), \tag{2}$$

where the near-optimal policies and action-value functions are now functions of the reward parameters ψ_i and dynamics parameters μ_i, in addition to their standard inputs. Notably, this formulation is closely related to prior works on goal-conditioned RL (Andrychowicz et al. 2017; Schroecker and Isbell 2020), and universal value function approximators (UVFA) (Schaul et al. 2015; Borsa et al. 2018).

Consequently, our problem is formally defined as approximating the mapping shown in Equation (2) to obtain the approximated near-optimal action-value function $\hat{Q}_\phi(s, a|\psi, \mu)$ and policy $\hat{\pi}_\theta(a|s, \psi, \mu)$ that are parameterized by ϕ and θ, respectively. Once such mapping is obtained, one can predict and observe near-optimal trajectories by rolling out the approximated policy $\hat{\pi}_\theta$ without necessarily training the RL solver from scratch:

$$\hat{\pi}_\theta(a|s, \psi, \mu) \xrightarrow{\text{Policy Rollout in the Environment}} \hat{\tau}(\psi, \mu), \tag{3}$$

where $\hat{\tau}(\psi, \mu)$ is the near-optimal trajectory corresponding to the reward parameters ψ and dynamics parameters μ.

3.2 Generating Optimal Policies and Optimal Value Functions with Hypernetworks

Our goal is to approximate the mapping described in Equation (2). To that end, we assume having access to a family of near-optimal policies π_i^* that were trained independently on instances of $M_i \in \mathscr{M}$. A dataset of near-optimal trajectories is then collected by rolling out each π_i^* on its corresponding MDP $\mathcal{M}_i$. Thus, samples are drawn from the stationary state distribution of the near-optimal policy $d^{\pi^*}(s)$.

Consequently, the inputs to the learner are tuples of states, reward parameters and dynamics parameters $\langle s, \psi_i, \mu_i \rangle$ and the targets are tuples of near-optimal actions and action-values $\langle a^*, q^* \rangle$. We can frame the approximation problem of Equation (2) as a supervised learning problem under the following conditions:

Assumption 1. The parameters of the reward function R_ψ and transition dynamics $\mathcal{T}_\mu$ are sampled independently and identically from distributions over the parameters $\psi_i \sim p(\psi)$ and $\mu_i \sim p(\mu)$, respectively.

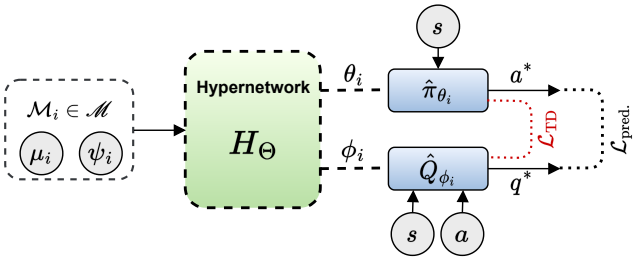


Figure 2: Diagram of our learning framework for universal approximation of RL solutions. Given reward parameters ψ_i and dynamics parameters μ_i, the hypernetwork H_Θ generates weights of the approximated near-optimal policy $\hat{\pi}_\theta$ and value function $\hat{Q}_\phi$. The only learnable parameters are Θ.

Assumption 2. The RL algorithm that is to be approximated, as shown in Equations (1) and (2), is converged to the near-optimal value function and policy.

Assumption 1 is a common assumption on the task distribution in meta-learning methods (Finn, Abbeel, and Levine 2017). While Assumption 2 appears strong, it is related to the common assumption made in imitation learning where the learner has access to expert demonstrations (Ross, Gordon, and Bagnell 2011; Ho and Ermon 2016). Nevertheless, we empirically show that Assumption 2 can be relaxed to an extent in practice, while still achieving strong zero-shot performance, as shown in Section 4.

Importantly, we assume no prior knowledge on the structure of the RL algorithm nor on the nature (stochastic or deterministic) of the policy that were used to generate the data. Notably, since the optimal policy of a given MDP is deterministic (Puterman 2014), we can parameterize the approximated near-optimal policy $\hat{\pi}_\theta$ as a deterministic function $\hat{\pi}_\theta : \mathcal{S} \to \mathcal{A}$, without any loss of optimality.

We propose to use hypernetworks (Ha, Dai, and Le 2016) for solving this approximation problem. Conditioned on the parameters ψ_i, μ_i of an MDP $\mathcal{M}_i \in \mathscr{M}$, as shown in Figure

Algorithm 1: HyperZero

Inputs: Parameterized reward function R_ψ and transition dynamics $\mathcal{T}_\mu$, distribution $p(\psi)$ and $p(\mu)$ over parameters, hypernetwork H_Θ, main networks $\hat{\pi}_\theta$ and $\hat{Q}_\phi$.
Hyperparameters: RL algorithm, learning rate α of hypernetwork H_Θ, number of tasks N.

Initialize dataset $\mathcal{D}$ of near-optimal trajectories
for $i = 1$ to N **do**
    Sample MDP $\mathcal{M}_i \in \mathscr{M}$: $\psi_i \sim p(\psi), \mu_i \sim p(\mu_i)$
    Obtain π_i^* and Q_i^* of $\mathcal{M}_i \in \mathscr{M}$ with an RL solver
    Store near-optimal trajectories τ_i^*: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_i^*\}$
end for
while not done **do**
    Sample mini-batch $\langle \psi_i, \mu_i, s, a^*, s', r, q^* \rangle \sim \mathcal{D}$
    Generate $\hat{\pi}_{\theta_i}$ and $\hat{Q}_{\phi_i}$: $[\theta_i; \phi_i] = H_\Theta(\psi_i, \mu_i)$
    $\Theta \leftarrow \text{argmin}\, \mathcal{L}_{\text{pred.}}(\Theta) + \mathcal{L}_{\text{TD}}(\Theta)$ ▷ Eqn. (4-5)
end while

2, the hypernetwork H_Θ generates weights of the approximated near-optimal policy $\hat{\pi}_\theta$ and action-value function $\hat{Q}_\phi$. Following the literature on hypernetworks, we refer to the generated policy and value networks as *main* networks.

Consequently, the hypernetwork is trained via minimizing the error for predicting the near-optimal action and values by forward passing the main networks:

$$\mathcal{L}_{\text{pred.}}(\Theta) = \mathbb{E}_{(\psi_i,\mu_i,s,a^*,q^*)\sim\mathcal{D}}\Big[(\hat{Q}_{\phi_i}(s,a^*) - q^*)^2\Big] + \mathbb{E}_{(\psi_i,\mu_i,s,a^*)\sim\mathcal{D}}\Big[(\hat{\pi}_{\theta_i}(s) - a^*)^2\Big] \quad (4)$$

where $[\theta_i;\phi_i] = H_\Theta(\psi_i,\mu_i)$ and $\mathcal{D}$ is the dataset of near-optimal trajectories collected from the family of MDPs $\mathscr{M}$. Notably, this training paradigm effectively decouples the problem of learning optimal values/actions from the problem of learning the mapping of MDP parameters to the space of optimal value functions and policies. Thus, as observed in other works on hypernetworks (Ha, Dai, and Le 2016; Galanti and Wolf 2020; von Oswald et al. 2019; Faccio et al. 2022a), this level of modularity results in a simplified and more efficient learning.

3.3 Temporal Difference Regularization

A key challenge in using supervised learning approaches for function approximation in deep RL is the temporal correlation existing within the samples, which results in the violation of the i.i.d. assumption. Common practices in deep RL for stabilizing the learning is to use a target network to estimate the temporal difference (TD) error (Lillicrap et al. 2015; Mnih et al. 2013). In this paper, we propose a novel regularization technique based on the TD loss to stabilize the training of the hypernetwork for zero-shot transfer learning.

As stated in Assumption 2, we assume having access to near-optimal RL solutions that were generated from a converged RL algorithm. As a result, our framework differs from the works on imitation learning (Ross, Gordon, and Bagnell 2011; Bagnell 2015; Ho and Ermon 2016) since samples satisfy the Optimal Bellman equation of the underlying MDP $\mathcal{M}_i \in \mathscr{M}$ and, more importantly, we have access to the near-optimal action-values q^* for a given transition sample $\langle s, a^*, s', r\rangle$.

Therefore, we propose to use the TD loss to regularize the approximated critic $\hat{Q}_\phi$ by moving the predicted target value towards the current value estimate, which is obtainable from the ground-truth RL algorithm:

$$\mathcal{L}_{\text{TD}}(\Theta) = \mathbb{E}_{(\psi_i,\mu_i,s,a^*,s',r,q^*)\sim\mathcal{D}}\Big[(r + \gamma\hat{Q}_{\phi_i}(s',\overline{a'}) - q^*)^2\Big] \quad (5)$$

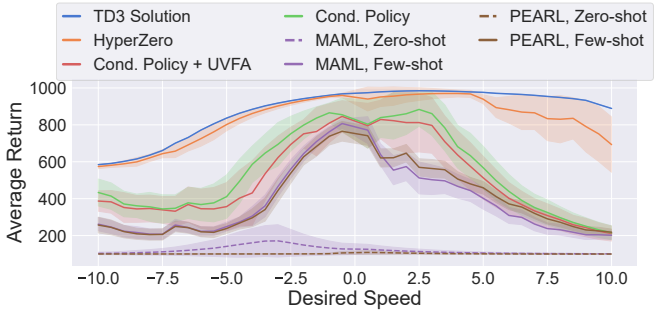
where $\overline{a'}$ is obtained from the approximated deterministic policy $\hat{\pi}_\theta(s')$ with stopped gradients. Note that our application of the TD loss differs from that of standard function approximation in deep RL (Mnih et al. 2013; Lillicrap et al. 2015); instead of moving the current value estimate towards the target estimates, our TD loss moves the target estimates towards the current estimates. While this relies on Assumption 2, we show that in practice applying the TD loss is beneficial as it enforces the approximated policy and critic to be consistent with respect to the Bellman equation. Algorithm 1 shows the pseudo-code of our learning framework.

4 Evaluation

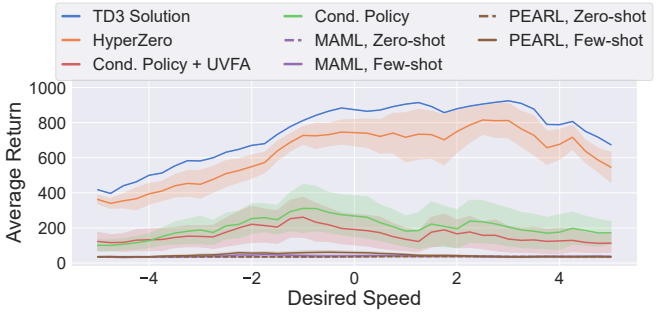
We evaluate our proposed method, referred to as *HyperZero* (**hyper**networks for **zero**-shot transfer) on a series of challenging continuous control tasks from DeepMind Control Suite. The primary goal in our experiments is to study the zero-shot transfer ability of the approximated RL solutions to novel dynamics and rewards settings.

4.1 Experimental Setup

Environments. We use three challenging environments for evaluation: cheetah, walker, and finger. For an easier visualization and realization of reward parameters, in all cases the reward parameters correspond to the desired speed of the motion which consists of both negative (moving backward) and positive (moving forward) values. Depending on the environment, dynamics changes correspond to changes in a body size and its weight/inertia. Full details of the environments and their parameters are in Appendix A.

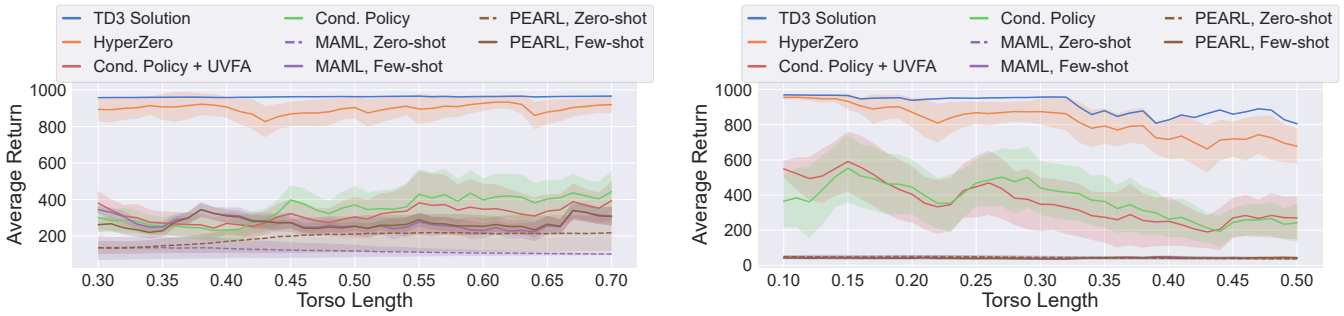


(a) Cheetah environment.



(b) Walker environment.

Figure 3: Zero-shot transfer to new reward settings on DM control environments, obtained on 5 seeds for random split of train/test tasks. Solid lines present the mean and shaded regions present the standard deviation of the average return across the seeds. Horizontal axis shows the desired speed, which is a function of the reward parameters ψ_i.



(a) Cheetah environment.

(b) Walker environment.

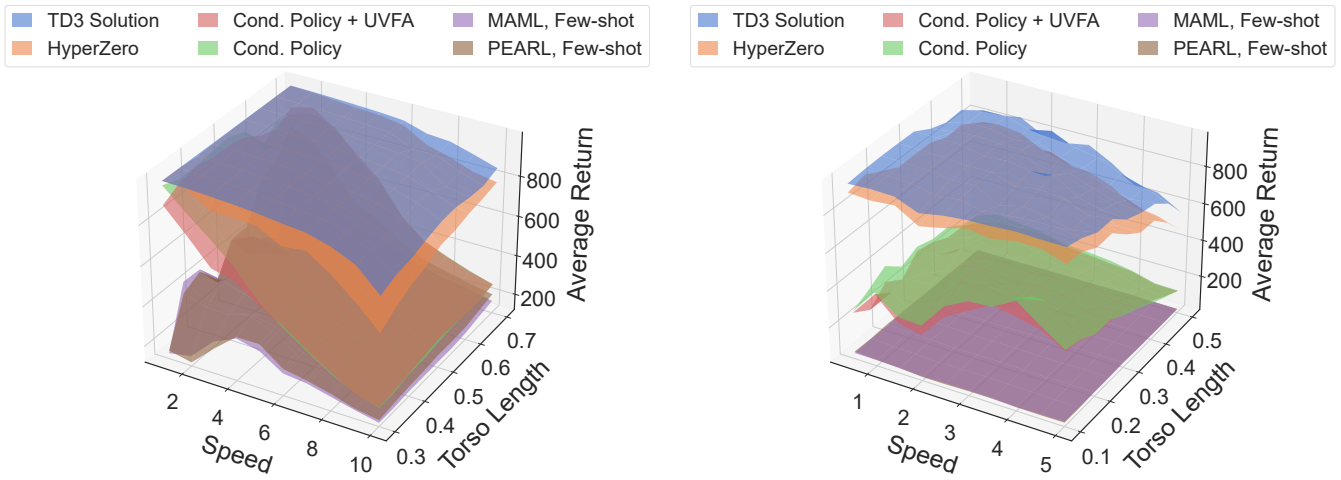
Figure 4: Zero-shot transfer to new dynamics settings on DM control environments, obtained on 5 seeds for random split of train/test tasks. Solid lines present the mean and shaded regions present the standard deviation of the average return across the seeds. Horizontal axis shows the value of dynamics parameter μ_i; that is torso length for the cheetah and walker, and finger length for the finger. Notably, a change in the shape of the geometry results in changes in the weight and inertia parameters.

RL Training and Dataset Collection. We use TD3 (Fujimoto, Hoof, and Meger 2018) as the RL algorithm that is to be approximated. Each MDP $\mathcal{M}_i \in \mathscr{M}$, generated by sampling $\psi_i \sim p(\psi)$ and $\mu_i \sim p(\mu)$, is used to independently train a standard TD3 agent on proprioceptive states for 1 million steps. Consequently, the final solution is used to generate 10 rollouts to be added to the dataset $\mathcal{D}$. Learning curves for the RL solutions are in Appendix B.3. As these results show, in some instances, the RL solution is not fully converged after 1 million steps. Despite this, HyperZero is able to approximate the mapping reasonably well, thus indicating Assumption 2 can be relaxed to an extent in practice.

Train/Test Split of the Tasks. To reliably evaluate the zero-shot transfer abilities of HyperZero to novel reward/-dynamics settings against the baselines, and to rule out the possibility of selective choosing of train/test tasks, we randomly divide task settings into train ($\%85$) and test ($\%15$) sets. We consequently report the mean and standard deviation of the average return obtained on 5 seeds.

Baselines. We compare HyperZero against common baselines for multitask and meta learning:

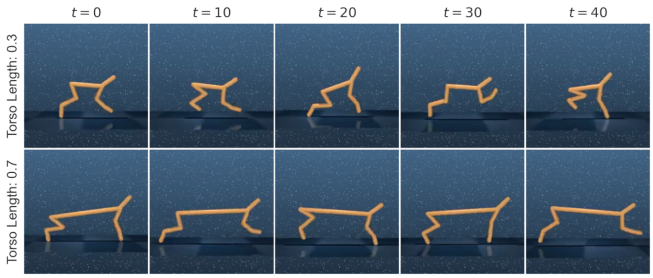
1. Context-conditioned policy; trained to predict actions, similarly to imitation learning methods.
2. Context-conditioned policy paired with UVFA (Schaul et al. 2015); trained to predict actions and values. It further benefits from using our proposed TD loss $\mathcal{L}_{\text{TD}}$, similarly to HyperZero.
3. Context-conditioned meta policy; trained with MAML



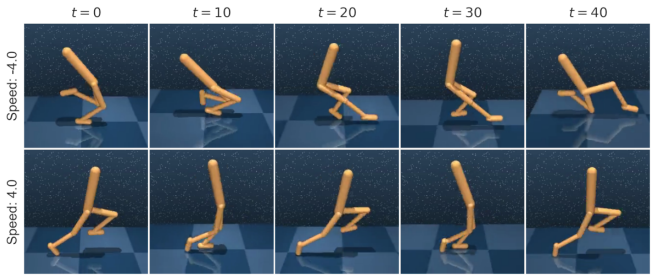
(a) Cheetah environment.

(b) Walker environment.

Figure 5: Zero-shot transfer to new reward and dynamics settings on DM control environments, obtained on 5 seeds for random split of train/test tasks. Each surface present the mean of the average return across the seeds. X-axis shows the desired speed, which is a function of the reward parameters ψ_i, while Y-axis shows the value of the dynamics parameter μ_i. The surfaces are smoothed for visual clarity. 2D plots of these 3D diagrams are presented in Appendix B.1 for better comparison.



(a) Cheetah environment with different torso lengths.



(b) Walker environment with different desired speeds.

Figure 6: Rollout of a trained HyperZero on different task parameters. (a) The trained HyperZero is used to rollout the cheetah environment with torso lengths of 0.3 and 0.7. (b) The trained HyperZero is used to rollout the walker environment with desired speeds of -4 and +4. Additional results are in Appendix B.2.

(Finn, Abbeel, and Levine 2017) to predict actions and evaluated for both zero-shot and few-shot transfer. Our context-conditioned meta-policy can be regarded as an adaptation of PEARL (Rakelly et al. 2019) in which the inferred task is substituted by the ground-truth task.

4. PEARL (Rakelly et al. 2019) policy; trained to predict actions. Unlike other baselines, PEARL does not assume access to the MDP context and instead it infers the the context from states and actions.

Notably, since MAML and PEARL are known to perform poorly for zero-shot transfer, we evaluate the meta policy for both zero-shot and few-shot transfers. In the latter, prior to evaluation, the meta policy is finetuned with near-optimal trajectories of the test MDP $\mathcal{M}_i$ generated by the actual RL solution.

Finally, for a fair comparison with hypernetworks, all methods follow the same two-stage training paradigm described in Section 3.1, have a learnable task embedding, and share the same network architecture. Full implementation details are in Appendix C.

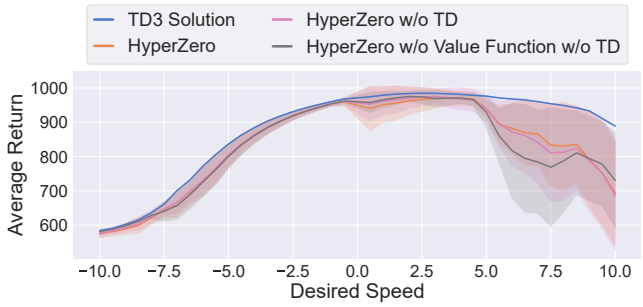
4.2 Results

Zero-shot Transfer. We compare the zero-shot transfer of HyperZero against the baselines in the three cases of changed rewards, changed dynamics, and simultaneously changed rewards and dynamics; results are shown in Figures 3, 4, and 5, respectively. Additional results are in Appendix B.1. As suggested by these results, in all environments and transfer scenarios, HyperZero significantly outperforms the baselines, demonstrating the effectiveness of our learning framework for approximating an RL algorithm as a mapping from a parameterized MDP $\mathcal{M}_i$ to a near-optimal policy π_i^* and action-value function Q_i^*.

Importantly, the context-conditioned policy (paired with UVFA) consists of all the major components of HyperZero, including near-optimal action and value prediction, and TD regularization. As a result, the only difference is that HyperZero learns to generate policies conditioned on the context which is in turn used to predict actions, while the context-conditioned policy learns to predict actions conditioned on the context. We hypothesize two main reasons for the significant improvements gained from such use of hypernetworks in our setting. First, aligned with similar observations in the literature (Galanti and Wolf 2020; von Oswald et al. 2019), hypernetworks allow for effective abstraction of the learning problem into two levels of policy (or equivalently value function) generation and action (or equivalently value) prediction.

Second, as hypernetworks are used to learn the mapping from MDP parameters to the space of policies, that is $(\psi_i, \mu_i) \rightarrow \pi_i^*$, they achieve generalization across the space of policies. On the contrary, since the context-conditioned policy simultaneously learns the mapping of states and MDP parameters to actions, that is $(s, \psi_i, \mu_i) \rightarrow a^*$, it is only able to achieve generalization over the space of actions, as opposed to the more general space of policies.

Finally, due to the strong zero-shot transfer ability of the approximated solution to new rewards and dynamics, one can use it to visualize the near-optimal trajectory τ_i^* for novel tasks without necessarily training the RL algorithm. A possible application of this approach would be for task visualization or environment design, as well as manual reward



(a) Cheetah environment.

Figure 7: Ablation study on the improvements gained from generating the optimal value function and using the TD loss. Results are obtained on 5 seeds for random split of train/test tasks. Solid lines present the mean and shaded regions present the standard deviation of the average return across the seeds. Horizontal axis shows the desired speed, which is a function of the reward parameters ψ_i.

shaping. As an example, Figure 6 shows sample trajectories generated by rolling out trained HyperZero models conditioned on different reward/dynamics parameters. Additional trajectories are in Appendix B.2.

Ablation Study on HyperZero Variants. In Figure 7, we carry out an ablation study on the improvements gained from generating the near-optimal value function and using our proposed TD loss from Equation (5). We draw two conclusions from this study; first, generating the action-value function Q_i^* alongside the policy π_i^* provides additional learning signal for training the hypernetwork. Furthermore, incorporating the TD loss between the generated policy and action-value function ensures the two generated networks are consistent with one another with respect to the Bellman equation and results in overall better performance and generalization.

While the improvements may appear to be small, we suspect that gains would be larger in visual control problems, as generating the value function will provide a rich learning signal for representation learning. More importantly, the generated value function can have other applications, such as beings used in policy gradient methods for further training the generated policy with environment interactions (offline-to-online RL) (Lee et al. 2022). While this is left for future work, we wanted to ensure that our framework is capable of generating the value function alongside the policy.

5 Related Work

The robustness and generalization of behaviors has long been studied in control and RL.

Transfer, Contextual and Meta RL. Past work has studied numerous forms of Transfer Learning (Taylor and Stone 2009), where MDP components including the state space, action space, dynamics or reward are modified between the training conducted on one or many source tasks, prior to performance on one or more targets. Depending on the learner's view of sources and targets, the problem is called contextual policy search (Kupcsik et al. 2017) life-long learning (Abel et al. 2018), curriculum learning (Portelas et al. 2020), or meta learning (Finn, Abbeel, and Levine 2017), but our particular variant, with an always-observable parameter vector and no chance to train or fine-tune on the target is most aptly named zero-shot contextual RL. Within that problem, a common concern has been how to interpolate in the space of contexts (equivalent to our parameters), while preserving details of the policy-space solution (Barbaros et al. 2018). This is precisely where the power of our hypernetwork architecture extends prior art.

Hypernetworks in RL. While hypernetworks (Ha, Dai, and Le 2016) have been used extensively in supervised learning problems (von Oswald et al. 2019; Galanti and Wolf 2020; Krueger et al. 2017; Zhao et al. 2020), their application to RL algorithms remains relatively limited. Recent work of Sarafian, Keynan, and Kraus (2021) use hypernetworks to improve gradient estimation of Q functions and policy networks in policy gradient algorithms. In multi-agent RL, hypernetworks are used to generate policies or value functions based on agent properties (Rashid et al. 2018; Iqbal et al. 2020, 2021; de Witt et al. 2020; Zhou et al. 2020). Furthermore, hypernetworks have been used to model an evolving dynamical system in continual model-based RL (Huang et al. 2021). Related to our approach, Faccio et al. (2022a) use hypernetworks to learn goal-conditioned optimal policies; the key distinguishing factor of our approach is that we focus on zero-shot transfer across a family of MDPs with different reward and dynamics functions, while the method of Faccio et al. (2022a) aims to solve a single goal-conditioned MDP.

Upside Down RL. Upside down RL (UDRL) is a redefinition of the RL problem transforming it into a form of supervised learning. UDRL, rather than learning optimal policies using rewards, teaches agents to follow commands. This method maps input observations as commands to action probabilities with supervised learning conditioned on past experiences (Srivastava et al. 2019; Schmidhuber 2019). Related to this idea are offline RL models that use sequence modeling as opposed to supervised learning to model behavior (Janner, Li, and Levine 2021; Chen et al. 2021). Similarly to UDRL, many RL algorithms incorporate the use of supervised learning in their model (Schmidhuber 2015; Rosenstein et al. 2004). One such technique is hindsight RL in which commands correspond to goal conditions (Andrychowicz et al. 2017; Rauber et al. 2017; Harutyunyan et al. 2019). Another approach is to use forward models as opposed to the backward ones used in UDRL (Arjona-Medina et al. 2019). Recently, Faccio et al. (2022a) propose a method that evaluates generated polices in the command space rather than optimizing a single policy for achieving a desired reward.

6 Conclusion

This paper has described an approach, named HyperZero, which learns to generalize optimal behavior across a family of tasks. By training on the full RL solutions of training tasks, including their optimal policy and value function parameters, the hypernetworks used in our architecture are trained to directly output the parameters of complex neural network policies capable of solving unseen target tasks. This work extends the performance of zero-shot generalization over prior approaches. Our experiments demonstrate that our zero-shot behaviors achieve nearly full performance, as defined by the performance of the optimal policy recovered by an RL learner training for a large amount of iterations on the target task itself.

Due to the strong generalization of our method, with minimal test-time computational requirements, our approach is suitable for deployment in live systems. We also highlight the opportunity for human-interfaces and exploration of RL solutions. In short, this new level of rapid, but powerful, general behavior can provide significant opportunity for the practical deployment of RL-learned behavior in the future.

References

Abel, D.; Jinnai, Y.; Guo, Y.; Konidaris, G.; and Littman, M. L. 2018. Policy and Value Transfer in Lifelong Rein-

forcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; and Zaremba, W. 2017. Hindsight experience replay. *Advances in neural information processing systems*, 30.

Arjona-Medina, J. A.; Gillhofer, M.; Widrich, M.; Unterthiner, T.; Brandstetter, J.; and Hochreiter, S. 2019. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32.

Bagnell, J. A. 2015. An invitation to imitation. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst.

Barbaros, V.; van Hoof, H.; Abdolmaleki, A.; and Meger, D. 2018. Eager and Memory-Based Non-Parametric Stochastic Search Methods for Learning Control. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Barreto, A.; Hou, S.; Borsa, D.; Silver, D.; and Precup, D. 2020. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48): 30079–30087.

Bellman, R. 1966. Dynamic programming. *Science*, 153(3731): 34–37.

Borsa, D.; Barreto, A.; Quan, J.; Mankowitz, D. J.; van Hasselt, H.; Munos, R.; Silver, D.; and Schaul, T. 2018. Universal Successor Features Approximators. In *International Conference on Learning Representations*.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34: 15084–15097.

de Witt, C. S.; Peng, B.; Kamienny, P.-A.; Torr, P.; Böhmer, W.; and Whiteson, S. 2020. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709*.

Faccio, F.; Herrmann, V.; Ramesh, A.; Kirsch, L.; and Schmidhuber, J. 2022a. Goal-Conditioned Generators of Deep Policies. *arXiv preprint arXiv:2207.01570*.

Faccio, F.; Ramesh, A.; Herrmann, V.; Harb, J.; and Schmidhuber, J. 2022b. General Policy Evaluation and Improvement by Learning to Identify Few But Crucial States. *arXiv preprint arXiv:2207.01566*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Galanti, T.; and Wolf, L. 2020. On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*, 33: 10409–10419.

Ha, D.; Dai, A.; and Le, Q. V. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Hallak, A.; Di Castro, D.; and Mannor, S. 2015. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*.

Harb, J.; Schaul, T.; Precup, D.; and Bacon, P.-L. 2020. Policy evaluation networks. *arXiv preprint arXiv:2002.11833*.

Harutyunyan, A.; Dabney, W.; Mesnard, T.; Gheshlaghi Azar, M.; Piot, B.; Heess, N.; van Hasselt, H. P.; Wayne, G.; Singh, S.; Precup, D.; et al. 2019. Hindsight credit assignment. *Advances in neural information processing systems*, 32.

Ho, J.; and Ermon, S. 2016. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29.

Huang, Y.; Xie, K.; Bharadhwaj, H.; and Shkurti, F. 2021. Continual model-based reinforcement learning with hypernetworks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 799–805. IEEE.

Iqbal, S.; de Witt, C. A. S.; Peng, B.; Böhmer, W.; Whiteson, S.; and Sha, F. 2020. Ai-qmix: Attention and imagination for dynamic multi-agent reinforcement learning. *arXiv preprint arXiv:2006.04222*.

Iqbal, S.; De Witt, C. A. S.; Peng, B.; Böhmer, W.; Whiteson, S.; and Sha, F. 2021. Randomized Entity-wise Factorization for Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*, 4596–4606. PMLR.

Janner, M.; Li, Q.; and Levine, S. 2021. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286.

Jiang, N.; Krishnamurthy, A.; Agarwal, A.; Langford, J.; and Schapire, R. E. 2017. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, 1704–1713. PMLR.

Krueger, D.; Huang, C.-W.; Islam, R.; Turner, R.; Lacoste, A.; and Courville, A. 2017. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*.

Kupcsik, A.; Deisenroth, M. P.; Peters, J.; Loh, A. P.; Vadakkepat, P.; and Neumann, G. 2017. Model-based contextual policy search for data-efficient generalization of robot skills.

Lee, S.; Seo, Y.; Lee, K.; Abbeel, P.; and Shin, J. 2022. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, 1702–1712. PMLR.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Portelas, R.; Colas, C.; Hofmann, K.; and Oudeyer, P.-Y. 2020. Teacher algorithms for curriculum learning of Deep

RL in continuously parameterized environments. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 100, 835–853.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rakelly, K.; Zhou, A.; Finn, C.; Levine, S.; and Quillen, D. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, 5331–5340. PMLR.

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, 4295–4304. PMLR.

Rauber, P.; Ummadisingu, A.; Mutz, F.; and Schmidhuber, J. 2017. Hindsight policy gradients. *arXiv preprint arXiv:1711.06006*.

Rosenstein, M. T.; Barto, A. G.; Si, J.; Barto, A.; Powell, W.; and Wunsch, D. 2004. Supervised actor-critic reinforcement learning. *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*, 359–380.

Ross, S.; Gordon, G.; and Bagnell, D. 2011. 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*, 627–635. JMLR Workshop and Conference Proceedings.

Sarafian, E.; Keynan, S.; and Kraus, S. 2021. Recomposing the reinforcement learning building blocks with hypernetworks. In *International Conference on Machine Learning*, 9301–9312. PMLR.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal value function approximators. In *International conference on machine learning*, 1312–1320. PMLR.

Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural networks*, 61: 85–117.

Schmidhuber, J. 2019. Reinforcement Learning Upside Down: Don't Predict Rewards–Just Map Them to Actions. *arXiv preprint arXiv:1912.02875*.

Schroecker, Y.; and Isbell, C. 2020. Universal value density estimation for imitation learning and goal-conditioned reinforcement learning. *arXiv preprint arXiv:2002.06473*.

Sorg, J.; Lewis, R. L.; and Singh, S. 2010. Reward design via online gradient ascent. *Advances in Neural Information Processing Systems*, 23.

Srivastava, R. K.; Shyam, P.; Mutz, F.; Jaśkowski, W.; and Schmidhuber, J. 2019. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; and Liu, C. 2018. A survey on deep transfer learning. In *International conference on artificial neural networks*, 270–279. Springer.

Tang, Z.; Feng, Y.; and Liu, Q. 2022. Operator Deep Q-Learning: Zero-Shot Reward Transferring in Reinforcement Learning. *arXiv preprint arXiv:2201.00236*.

Taylor, M. E.; and Stone, P. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10(56): 1633–1685.

Teh, Y.; Bapst, V.; Czarnecki, W. M.; Quan, J.; Kirkpatrick, J.; Hadsell, R.; Heess, N.; and Pascanu, R. 2017. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30.

von Oswald, J.; Henning, C.; Grewe, B. F.; and Sacramento, J. 2019. Continual learning with hypernetworks. In *International Conference on Learning Representations*.

Xu, Z.; van Hasselt, H. P.; Hessel, M.; Oh, J.; Singh, S.; and Silver, D. 2020. Meta-gradient reinforcement learning with an objective discovered online. *Advances in Neural Information Processing Systems*, 33: 15254–15264.

Xu, Z.; van Hasselt, H. P.; and Silver, D. 2018. Meta-gradient reinforcement learning. *Advances in neural information processing systems*, 31.

Zhao, D.; Kobayashi, S.; Sacramento, J.; and von Oswald, J. 2020. Meta-Learning via Hypernetworks. In *4th Workshop on Meta-Learning at NeurIPS 2020 (MetaLearn 2020)*. NeurIPS.

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 *International Conference on Machine Learning*, 11436–11446. PMLR.

Zheng, Z.; Oh, J.; and Singh, S. 2018. On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems*, 31.

Zhou, M.; Liu, Z.; Sui, P.; Li, Y.; and Chung, Y. Y. 2020. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 11853–11864.