# Same State, Different Task: Continual Reinforcement Learning without Interference

**Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, Stephen J. Roberts**

University of Oxford
{skessler, jackph, ball, zohren, sjrob}@robots.ox.ac.uk

## Abstract

Continual Learning (CL) considers the problem of training an agent sequentially on a set of tasks while seeking to retain performance on all previous tasks. A key challenge in CL is catastrophic forgetting, which arises when performance on a previously mastered task is reduced when learning a new task. While a variety of methods exist to combat forgetting, in some cases tasks are fundamentally incompatible with each other and thus cannot be learnt by a single policy. This can occur, in reinforcement learning (RL) when an agent may be rewarded for achieving *different goals* from the *same observation*. In this paper we formalize this "interference" as distinct from the problem of forgetting. We show that existing CL methods based on single neural network predictors with shared replay buffers fail in the presence of interference. Instead, we propose a simple method, OWL, to address this challenge. OWL learns a factorized policy, using *shared* feature extraction layers, but *separate* heads, each specializing on a new task. The separate heads in OWL are used to prevent interference. At test time, we formulate policy selection as a multi-armed bandit problem, and show it is possible to select the best policy for an *unknown task* using feedback from the environment. The use of bandit algorithms allows the OWL agent to constructively re-use different continually learnt policies at different times during an episode. We show in multiple RL environments that existing replay based CL methods fail, while OWL is able to achieve close to optimal performance when training sequentially.

## 1 Introduction

Reinforcement Learning (RL (Sutton, Barto et al. 1998)) considers the problem of an agent taking sequential actions in an environment to maximize some notion of reward. In recent times there has been tremendous success in RL, with impressive results in Games (Silver et al. 2016) and Robotics (OpenAI et al. 2018), and even real-world settings (Bellemare et al. 2020). However, these successes have predominantly focused on learning a *single* task, with agents often brittle to changes in the dynamics or rewards (or even the seed (Henderson et al. 2018)).

One of the most appealing qualities of RL agents is their ability to continue to learn and thus improve throughout their lifetime. As such, there has recently been an increase in interest in *Continual Reinforcement Learning* (CRL), (Ring 1994; Khetarpal et al. 2020), a paradigm where agents train on tasks sequentially, while seeking to maintain performance on previously mastered tasks (Kirkpatrick et al. 2016; Schwarz et al. 2018; Rolnick et al. 2019). A key issue when training on sequential tasks is *catastrophic forgetting*, a consequence of gradient based learning, as training on a new task overwrites parameters which were important for previous tasks (Robert M. French 1999). While existing methods address this, they typically only consider the setup where each "task" is an entirely different RL environment, for example different games from the Arcade Learning Environment (Bellemare et al. 2012) or different simulated robotics environments (Ahn et al. 2019).

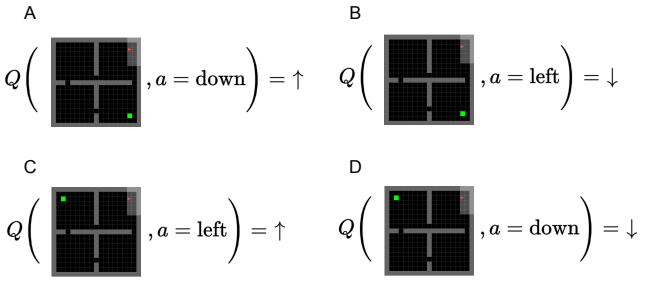


Figure 1: A and B a simple four rooms environment task where the goal (the green square) is located at the bottom right. C and D a different four rooms task where the goal is at a different location, the optimal Q-function should give different values for the same action for the same starting state. The symbol ↑ denotes a high Q-value and ↓ a low Q-value. Task agnostic, single predictor continual RL methods will be sub-optimal compared to methods which use the task identifier to condition their policies like OWL.

In this paper, we focus on a more challenging problem. Instead of distinct environments as different tasks, we consider learning to solve different tasks with the *same state space*. Since these tasks may not only have different but even opposite optimal actions for the same observation, training on them sequentially causes what we call "interference" which can in turn *induce* forgetting, as the agent directly optimizes for an opposing policy. Interference can also lead to reduced performance for future tasks, as the agent cannot fully learn

to solve a new task given it's retained knowledge of previous opposing objectives. This problem regularly arises in real-world settings, whereby there are a multitude of tasks with reward functions which all share the same state space (e.g. a visual observation of the world). Since previous CRL methods used different environments as different tasks then the agents can learn that the different state spaces correspond to different optimal behaviors and so interference is rarely exhibited.

We begin by showing a simple supervised setting where it is impossible to solve interfering tasks, continually with a single predictor despite using strong CL techniques to prevent forgetting, Figure 2. This setting can occur in RL where we have different goals but the same observation for different tasks, see Figure 1. We therefore introduce a new approach to CRL, which we call C**O**ntinual RL **W**ithout Conf**L**ict or OWL. OWL makes use of *shared feature extraction layers*, while acting based on *separate independent policy heads*. The use of the shared layers means that the size of the model scales gracefully with the number of tasks. The separate heads act to model multi-modal objectives and not as a means to retain task specific knowledge, as they are implicitly used in CL. Task agnostic, single predictor CRL methods which use experience replay (Rolnick et al. 2019) will thus suffer from this interference. But have the advantage of not having to infer the task the agent is in to then solve it. Experience replay has been shown to be a strong CL strategy (Balaji et al. 2020).

In the presence of interference, task inference is now necessary and at test time, OWL adaptively selects the policy using a method inspired by multi-armed bandits. We demonstrate in a series of simple yet challenging RL problems that OWL can successfully deal with interference, where existing methods fail. To alleviate forgetting we consider simple weight space (Kirkpatrick et al. 2016) and functional regularizations (Hinton, Vinyals, and Dean 2015) as these have been repeatedly shown to be effective in CRL (Nekoei et al. 2021).

Our core contribution is to identify a challenging new setting for CRL and propose a simple approach to solving it. As far as we are aware, we are the first to consider a bandit approach for selecting policies for deployment, inferring unknown tasks. The power of this method is further demonstrated in a set of generalization experiments, where OWL is able to solve tasks it has never seen before, up to 6x more successfully than the experience replay baseline (Rolnick et al. 2019).

## 2 Related Work

**Continual Learning in Supervised Learning:** Continual Learning (CL) is a sequential learning problem. One approach to CL, referred to as *regularization approaches* regularizes a NN predictor's weights to ensure that new learning produces weights which are similar to previous tasks (Kirkpatrick et al. 2016; Nguyen et al. 2018; Zenke, Poole, and Ganguli 2017). Alternatively, previous task functions can be regularized to ensure that the functions mapping inputs to outputs are remembered (Li and Hoiem 2017). By contrast, *expansion approaches* add new neural resources to enable learning new tasks while preserving components for specific tasks (Rusu et al. 2016; Lee et al. 2020). *Memory approaches* replay data from previous tasks when learning the current task. This can be performed with a generative model (Shin et al. 2017). Or small samples from previous tasks (*memories*) (Lopez-Paz and Ranzato 2017; Aljundi et al. 2019b; Chaudhry et al. 2019). Meta-learning pre-training has been explored to augment continual learning (and *context-dependent targets* can allow interference but is not explored) (Caccia et al. 2020).

**Continual Learning in Reinforcement Learning:** The CL regularization method EWC has been applied to DQN (Mnih et al. 2015) to learn over a series of Atari games (Kirkpatrick et al. 2016). Both Progressive Networks (Rusu et al. 2016) and Progress and Compress (Schwarz et al. 2018) are applied to policy and value function feature extractors for an actor-critic approach. These methods are told when the task changes.

CLEAR leverages experience replay buffers (Lin 1992) only to prevent forgetting: by using an actor-critic with V-trace importance sampling (Espeholt et al. 2018) of past experiences from the replay buffer catastrophic forgetting can be overcome (Rolnick et al. 2019). CLEAR uses a single predictor NNs and all experience is stored in the replay buffer and thus CLEAR is not required to know when the task changes. However, interference is ignored as multi-task performance of the all environments is similar to the sum of performances of individual environments. Different selective experience replay strategies can be used for preserving performance on past tasks (Isele and Cosgun 2018). Alternatively, (Mendez, Wang, and Eaton 2020) learns a policy gradient model which factorizes into task specific parameters and shared parameters. OWL is more general as it can wrap around any RL algorithm and be used for discrete action spaces and continuous control settings and achieve better results.

A number of previous works have studied transfer in multi-task RL settings where the goals within an environment change (Barreto et al. 2016; Schaul et al. 2015). Our work is related to (Yu et al. 2020) which considers interference between gradients in a multi-task setting.

**Interference:** this problem is discussed in (Rolnick et al. 2019) and has been studied in multi-task (for example (Bishop and Svensén 2012; Lin et al. 2019)) and meta-learning (for example, (Rajendran, Irpan, and Jang 2020)). However, we believe we are the first to consider it in CRL, and that the current state-of-the-art methods lack an approach to tackle it. In particular, we note that existing replay based methods such as (Rolnick et al. 2019) fail to address this issue, as the experience replay buffer will contain tuples of the same state-action pairs but different rewards for different tasks. Thus, the agent will not converge, as we show later in our experiments.

## 3 Background

### 3.1 Reinforcement Learning

A Markov Decision Process (MDP, (Bellman 1957)) is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$. Here $\mathcal{S}$ and $\mathcal{A}$ are the sets of states and actions respectively, such that for $s_t, s_{t+1} \in \mathcal{S}$ and $a_t \in$

$\mathcal{A}$. $P(s_{t+1}|s_t, a_t)$ is the probability that the system/agent transitions from $s_t$ to $s_{t+1}$ given action $a_t$ and $R(a_t, s_t, s_{t+1})$ is a reward obtained by an agent transitioning from $s_t$ to $s_{t+1}$ via $a_t$. The discount factor is represented by $\gamma \in (0, 1)$. Actions $a_t$ are chosen using a policy $\pi$ that maps states to actions: $\mathcal{S} \rightarrow \mathcal{A}$. In this paper we consider MDPs with finite horizons $H$. The return from a state is defined as the sum of discounted future rewards $R_t = \sum_{i=t}^{H} \gamma^{(i-t)} r(s_i, a_i)$. In RL the objective is to maximize $J = \mathbb{E}_{a_i \sim \pi}[R_1|s_0]$ given an initial state $s_0$, sampled from the environment.

One approach to maximizing expected return is to learn an action-value function for each state-action pair: $Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_t \sim P, r_t \sim R, a_t \sim \pi}[\sum_t R_t]$. We parameterize the action-value function with a neural network, denoted $Q(s_t, a_t; \theta_i)$, with parameters $\theta_i$. We optimize $\theta_i$ to minimize the expected temporal difference error using gradient descent:

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s_t, a_t \sim \rho}[(y_i - Q(s_t, a_t; \theta_i))^2] \quad (1)$$

We use Q-learning (Watkins and Dayan 1992), giving $y_i = r_t + \gamma \text{max}_a Q(s_{t+1}, a; \theta_{i-1})$, and the parameters $\theta_{i-1}$ are target network parameters. The associated update does not require on-policy samples, so learning is off-policy using a replay buffer (Lin 1992), hence $\rho$ is an empirical distribution that represents samples from the buffer. For continuous action settings, we learn a policy $\pi_\phi : \mathcal{S} \rightarrow \mathcal{A}$ using a neural network parameterized by $\phi$. For discrete action settings, our policy follows the maximum Q-value: $\pi := \text{argmax}_a Q(s, a)$.

### 3.2 Continual Learning

*Continual learning* (CL) is a paradigm whereby an agent must learn a set of tasks sequentially, while maintaining performance across all tasks. This presents several challenges, in particular avoiding forgetting and efficiently allocating resources for learning new tasks. In CL, the model is shown $M$ tasks $\mathcal{T}_\tau$ sequentially, where $\tau = 1, \ldots M$. A task is defined as $\mathcal{T}_\tau = \{p(X), p(Y|X), \tau\}$ where $X$ and $Y$ are input and output random variables.

In practice a task is comprised of $i = 1 \ldots N_\tau$ inputs $x_i \in \mathbb{R}^d$ and outputs $y_i \in \mathbb{R}$ ($\mathbb{N} \subset \mathbb{R}$). The model will lose access to the training dataset for task $\mathcal{T}_\tau$, it will be continually evaluated on all previous tasks $\mathcal{T}_j$ for $j \leq \tau$. $\tau$ can be used as a task identifier informing the agent when to start training on a new task. For a comprehensive review of CL scenarios see (van de Ven and Tolias 2018). For RL the definition of a task is simply an MDP and task identifier: $\mathcal{T}_\tau = \{\mathcal{S}_\tau, \mathcal{A}_\tau, p_\tau(s_1), p_\tau(s_{t+1}|s_t, a_t), R_\tau(a_t, s_t, s_{t+1}), \tau\}$, and so the agent will no longer be able to interact with previous environments, but must ensure that it can remember how to solve all past tasks/environments.

## 4 Catastrophic Forgetting vs. Interference

Forgetting occurs when performance on old tasks is reduced while learning new tasks. On the other hand, interference occurs when two or more tasks are incompatible for the same model. We re-use these definitions from CLEAR (Rolnick et al. 2019). We observe this when the multi-task objectives are multi-modal and tasks share the same observation space but have different goals/objectives.

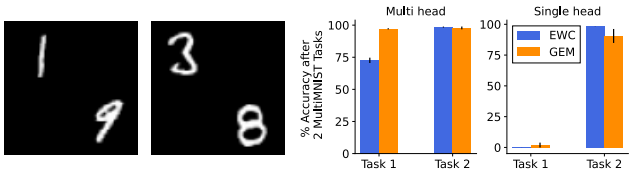


Figure 2: Left, two samples from the MultiMNIST dataset (Sabour, Frosst, and Hinton 2017). The first task $\mathcal{T}_1$ requires classifying the top left digit and the second task $\mathcal{T}_2$ requires classifying the bottom right. Right, results from continually learning on both MultiMNIST tasks. We compare two different CL methods EWC (Kirkpatrick et al. 2016) and GEM (Lopez-Paz and Ranzato 2017). Single head task agnostic setups suffer from catastrophic forgetting of the first task due to interference. The multi-head setup is a simple remedy to the problem of interference.

We demonstrate interference using the MultiMNIST dataset (Sabour, Frosst, and Hinton 2017), Figure 2. Each image is composed of two different MNIST digits and for $\mathcal{T}_1$ we are required to classify the top image and in $\mathcal{T}_2$ we are required to classify the bottom image. For both of these tasks the only difference is the objective. When we perform CL with a single predictor network or single-headed network with different CL strategies to alleviate forgetting we see that the interference between tasks causes almost $100\%$ forgetting of the first task, despite using established CL strategies. On the other hand using multi-headed networks allows us to model both objectives in MultiMNIST. We observe that we will get interference in the following.

**Observation 4.1.** *Consider two tasks $\mathcal{T}_i$ and $\mathcal{T}_j$. Let both tasks' input distributions $p_k(X)$ share the same support but have different conditional distributions $p_k(Y|X) = \mathcal{N}(f^k(X), \beta^{-1})$, where $f^k$ is a mean function with $f^i \neq f^j$ and $\beta^{-1}$ is data noise. Then the multi-task distribution is bi-modal and using a Gaussian likelihood will result in interference.*

This may seem contrived in the supervised setting, however, it is common throughout reinforcement learning. Consider a partially observable MDP (POMDP) where we receive an initial observation but do not know the goal location or reward function then an agent might require different policies for each task. We see an example of this in Figure 1 where in one task the goal is in the room below the agent and in the other the goal is in the room to the left. The most efficient policies will guide the agent in different directions depending on the task the agent is in. This observation has important consequences: methods which are task agnostic and do not condition on the task or do not use task specific parameters are susceptible to interference. Some CL methods use a single predictor and aim to approximate the multi-task setting by using storing samples in a buffer (Aljundi et al. 2019a; Rolnick et al. 2019). Furthermore single-headed networks are often used as more difficult CL scenarios when studying methods to mitigate forgetting (van de Ven and Tolias 2018; Farquhar and Gal 2018).

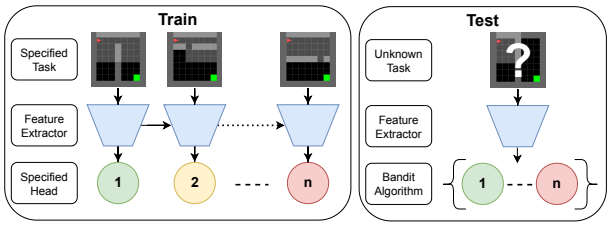


Figure 3: An overview of our approach. On the left we show the training setup. When training sequentially on different tasks we use the same feature extractor (blue), regularized with EWC, but maintain a set of distinct heads (green, yellow, ..., pink) corresponding to different tasks. On the right we show the test time adaptation. Note that the task is not known, and must be inferred through interaction with the environment.

Our solution is to model the multi-modality by learning a mixture of linear regressions (see: §14.5.1 (Bishop 2006)). The same applies to CRL: the Q-function needs to have separate weights for each task or needs to condition on the task to solve CRL environments with interference. In practice we will use multi-headed network predictors. *Whereas these are commonly employed to preserve previous task knowledge and prevent forgetting in CL, we are employing them as a means to prevent interference*. So our motivation for the use of multi-head networks is wholly different. Most supervised CL settings are benchmarked on vision tasks which use different distinct classes as tasks. Thus $p_\tau(X)$ and $p_\tau(Y|X)$ both change with task $\tau$; a single NN predictor can model this. However in RL only the reward function, $R_\tau(s_t, a_t, s_{t+1})$, need change with task $\tau$.

## 5 Continual RL without Conflict

At a high level, OWL uses an off-policy RL algorithm to train a Q-function across tasks sequentially. To prevent interference our key insight is that: 1.) we can use a single network with a shared feature extractor but multiple heads, parameterized by linear layers to fit individual tasks; 2.) we flush the experience replay buffer when starting to learn in a new task. At test time, we frame policy/head selection as a multi-armed bandit problem, to adaptively select the best policy. In this section we provide additional details on each component, describing first the structure of the Q-function, before moving to our test time adaptation.

### 5.1 Factorized Q-Functions

Multi-head networks are commonly used in CL (Li and Hoiem 2017; Nguyen et al. 2018), they enable learning task specific output mapping with a shared feature extractor. Multi-head networks are effective in that allow learning of task specific parameters which can we recalled and so help to alleviate forgetting. Single-head networks are commonly used as a more difficult baseline for CL benchmarks (Farquhar and Gal 2018; van de Ven and Tolias 2018). In our work multi-head networks are used as they prevent interference.

Algorithm 1: OWL: Training

**Input:** Tasks $\mathcal{T} = \{\mathcal{T}_i\}_{i=1}^{M}$.
**Initialize:** $\theta$ and $\phi$, $\Omega^Q = \Omega^\pi = \emptyset$.
**for** $t = 1, 2, \ldots, M$ **do**
  1. See Task $\mathcal{T}_t$
  2. Train Q-function with parameters $\{\theta_z, \theta_i\}$ and regularization $\Omega^Q$.
  **if** $\mathcal{A}$ *is continuous* **then**
    3. Train policy with parameters $\{\phi_z, \phi_i\}$ with regularization $\Omega^\pi$.
  4. Calculate Q-function EWC regularization and $\Omega^Q := \{\mathcal{L}^Q_{\text{EWC}}, \Omega^Q\}$.
  **if** $\mathcal{A}$ *is continuous* **then**
    5. Calculate policy EWC regularization and $\Omega^\pi := \{\mathcal{L}^\pi_{\text{EWC}}, \Omega^\pi\}$.
  6. Empty the experience reply buffer $\mathcal{D} = \emptyset$.
  7. Evaluate according to Algorithm 2.

**Alleviating forgetting:** We represent a factorized Q-function as having parameters $\theta = \{\theta_z, \theta_{1:M}\}$, where $\theta_z$ are feature extractor parameters and $\theta_{1:M}$ the heads. For discrete problems one can follow the maximum Q-values to obtain the next action. For parameterized policies we can equally construct our policy similarly with parameters $\phi = \{\phi_z, \phi_{1:M}\}$ where $\phi_z$ are the neural network feature extraction layers, and $\phi_{1:M}$ are linear policy heads. To address forgetting in the shared neural network feature extractors we use regularization methods. In particular we found EWC to work well and is a simple approach to prevent forgetting (Kirkpatrick et al. 2016), (we also tried a functional regularization (Hinton, Vinyals, and Dean 2015; Li and Hoiem 2017) but found it underperformed; see Section D). We train our agent according to Algorithm 1. As we see more and more tasks new heads can easily be added and so we do not need to prespecify the number of tasks or policy heads $M \in \{1, \ldots, \infty\}$.

### 5.2 Selecting Policies as a Multi-Armed Bandit Problem

At test time we do not tell OWL which task it is being evaluated on. We consider the set of arms $M$ to be the set of policies which can be chosen to act at each timestep of the test task. The aim is to find the policy which achieves the highest reward on a given test task. We use a modified version of the Exponentially Weighted Average Forecaster algorithm (Cesa-Bianchi and Lugosi 2006), as has been shown to be successful adapting components of RL algorithms (Ball et al. 2020). In this setup we consider $M$ experts making recommendations at the beginning of each round. After sampling a decision $i_t \in \{1, \cdots, M\}$ from a distribution $\mathbf{p}^t \in \Delta_M$ with the form $\mathbf{p}^t(i) \propto \exp(\ell_t(i))$ the learner experiences a loss $l^t_{i_t} \in \mathbb{R}$. The distribution $\mathbf{p}^t$ is updated by changing $\ell_t$ as follows:

$$\ell_{t+1}(i) = \begin{cases} \ell_t(i) + \eta \frac{l^t_i}{\mathbf{p}^t(i)} & \text{if } i = i_t \\ \ell_t(i) & \text{otherwise,} \end{cases} \quad (2)$$

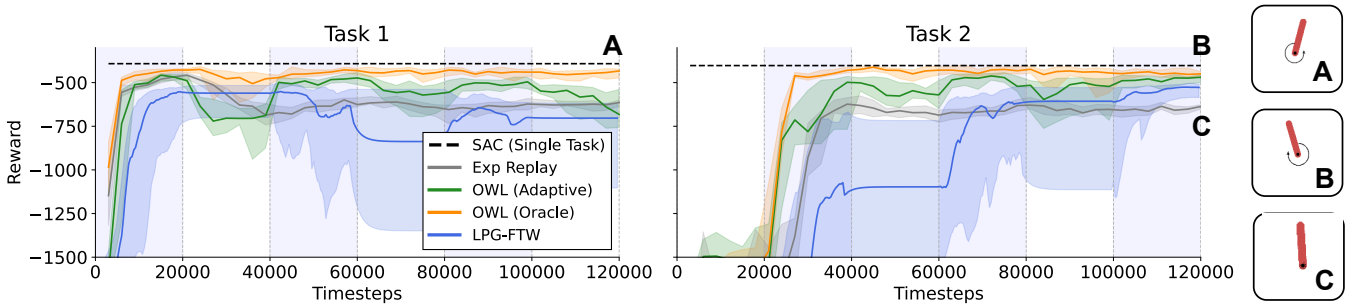


Figure 4: Median performance (with inter-quartile range) across 10 seeds. Pale blue shaded regions correspond to the timesteps when the task in question is being trained on. A, Task 1 arm position from the OWL agent (oracle) B, Task 2 arm position from the OWL agent (oracle) C, Optimal arm position for Exp Replay for Tasks 1 and 2 (note that the reward function also has angular velocity terms thus the OWL/SAC agent isn't able to place the arms exactly at $\pm 90^\circ$ without obtaining a sub optimal reward). Exp Replay clearly displays interference.

for some step size parameter $\eta$. We consider the case where the selection of $\phi_i$ is thought of as choosing among $M$ experts which we identify as the different policies $\{\phi_i\}_{i=1}^M$, trained on the corresponding Q-functions $\{\theta_i\}_{i=1}^M$. The loss we consider is of the form $l_{i_t} = 1/\hat{G}_{\phi_i}(\theta_i)$, where $G_{\phi_i}(\theta_i)$ is the TD error or the log likelihood of the observed reward from the test task, $r_t$ given the predicted Q-values. If required, we can then perform a normalization of $G$, hence $\hat{G}$. Henceforth we denote by $\mathbf{p}_\phi^t$ the exponential weights distribution over $\phi$ values at time $t$. The pseudocode for our test-time procedure is shown in Algorithm 2.

## 6 Experiments

To test our approach, we consider challenging CRL problems where tasks have similar or identical state and actions spaces but distinct goals/rewards. Our main hypothesis is that these tasks cannot be solved continually with a single policy, using a shared replay buffer. Our primary baseline, which we call Experience Replay (Exp Replay in figures) corresponds to this case (Rolnick et al. 2019) and has been shown to be a very effective baseline in CL (Balaji et al. 2020). In each setting we test two versions of our algorithm, which we refer to as Oracle and Adaptive. With the Oracle, the OWL agent is told at test time which task is being evaluated. Finally, we consider the multi-arm bandit (MAB) approach, denoted Adaptive. Code is available at https://github.com/skezle/owl.

### 6.1 Pendulums with Interfering Goals

The first setting we consider is a simple yet challenging take on the well-known Pendulum-v0 environment (Brockman et al. 2016). Typically, the policy is rewarded for placing the pendulum at $0^\circ$. Instead, we amend the reward function to produce two interfering tasks, with optimal positions: $\{+90^\circ, -90^\circ\}$. We have continuous actions and train a multi-head policy and Q-function using Soft Actor Critic (Haarnoja et al. 2018b). We train on each task three times, switching every $20,000$ environment steps. For more details on our implementation see Section C.1. Results are shown in Fig. 4.

Algorithm 2: OWL: Testing

**Input:** tasks seen so far $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_\tau\}$, Q-functions $\{\phi_i\}_{i=1}^M$, step size $\eta$, maximum number of timesteps $T$.
**Initialize:** $\mathbf{p}_\phi^1$ as a uniform distribution, $s_1$ as the initial state of the test task.
**for** $\mathcal{T}_j \in \mathcal{T}$ **do**
  **for** $t = 1, \ldots, T-1$ **do**
    1. Select $i_t \sim \mathbf{p}_\phi^t$, and set $\pi_{\text{test}} = \pi_{\phi_{i_t}}$.
    2. Take action $a_t \sim \pi_{\text{test}}(s_t)$, and receive reward $r_t$ and the next state $s_{t+1}$ from $\mathcal{T}_j$.
    3. Use Equation 2 to update $\mathbf{p}_\phi^t$ with $l_{i_t}^t = \hat{G}_{\phi_t}(\theta_{t+1})$

First, we see evidence confirming our first hypothesis that training with a shared replay buffer over all tasks leads to suboptimal performance on both tasks due to the interfering nature of the tasks (Exp Replay). The Exp Replay agent (grey) learns to place the pendulum at $0^\circ$. As we see on the bottom right, this balances the conflicting goals, but is suboptimal for both individual tasks (see Fig.4C). Secondly, the Oracle version of OWL (orange), which knows the task under evaluation at test time, performs well since two separate policies trained on the individual tasks (black, dashed) and displays minimal forgetting. Encouragingly we can almost achieve this same performance without informing the agent of the task index, using our adaptive mechanism (green). Our method also outperforms LPG-FTW (Mendez, Wang, and Eaton 2020) which uses 2.5x more gradient steps as our method builds on top of SAC which yields state-of-the-art results in continuous control. Regarding feedback to the algorithm, we explore the importance of the probabilistic networks in the Appendix, Section D.

### 6.2 MiniGrid Environments

MiniGrid is a challenging set of procedurally generated maze environments (Chevalier-Boisvert, Willems, and Pal 2018). Each environment is partially observable, with the agent only

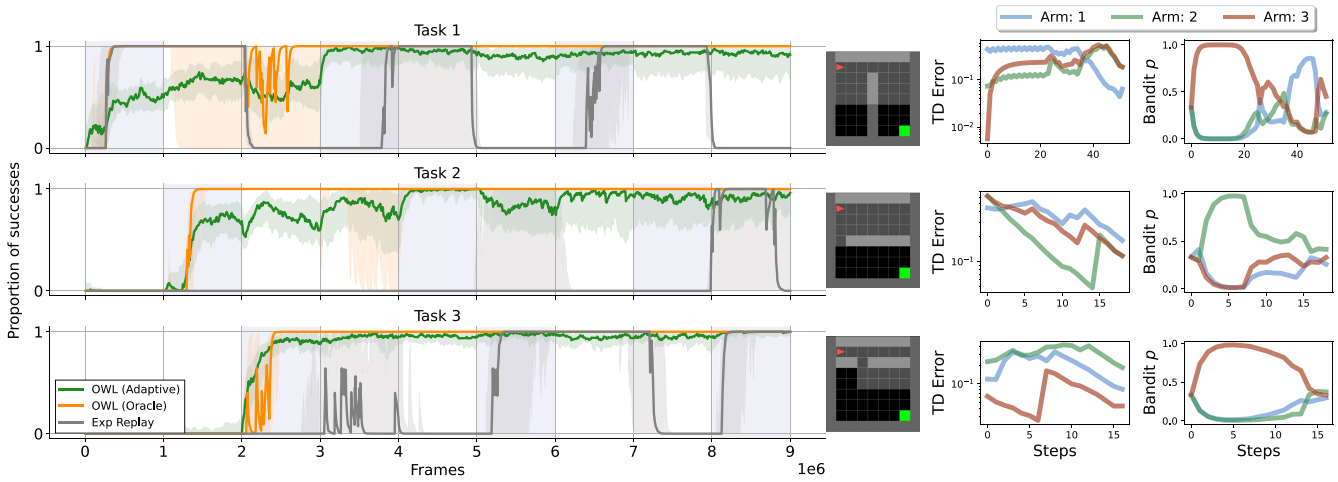


Figure 5: Left, Median performance across 10 seeds for three different MiniGrid environments trained continually. Shaded envelopes correspond to the inter-quartile range. Shaded pale blue regions corresponds to the current training task. OWL Adaptive and Oracle are able to prevent forgetting and interference while Exp. Replay fails. Right, Bandit arm probabilities over the course of a roll-out to demonstrate how the TD error feedback is used to select the right arm/policy to solve the task (note these have been smoothed for visualization purposes).

"seeing" a small region of visual input out of a larger state. Additionally, each state is an image, and rewards are sparse (the agent only receives a reward for navigating to the green tile in Figure 3), which makes it harder for agents to find learning signals. We use the SimpleCrossing environment, which has a single wall and gap. The environment seed corresponds to different wall position, orientation and gap. The initial observation can look identical (or very similar) for two different environments, and the agent has to explore to discover the wall location and door position.

We employ DQN to handle the discrete action space (Mnih et al. 2015), see Section C.2 for implementation details. We train the same methods as the previous experiment on three distinct MiniGrid grid worlds continually, repeating each three times for 1M steps. We use the TD error in Eq (1) as feedback to the MAB. OWL (Oracle) is able to consistently solve all environments after one round Figure 5. OWL (Adaptive) is able to dynamically select the correct policy most of the time after seeing each task once, and continuously improves, with the final performance almost matching OWL (oracle)[1]. Exp Replay exhibits significant interference between tasks.

**Scaling to more tasks.** We now scale to 5 SimpleCrossing tasks (denoted as SC in plots) and another set of 5 tasks with 3 SimpleCrossing and 2 DoorKey environments (denoted as SC+DK). The set of tasks are repeated 3 times each task is seen for 0.75M environment steps. For Exp Replay we adjust the buffer size to 4M ensure that data from all tasks are in the buffer over the course of training. We note that Exp. Replay again suffers from interference while OWL is able to overcome it (and forgetting) see Figures 10 and 11 in the appendix.

[1]For videos of OWL in action, see: https://sites.google.com/view/crlwithoutconflict/

We explore different arm selection strategies in Figure 6. We compare the multi-armed bandit (MAB) versus random policy head selection at each step of the roll-out (OWL (rand)), versus selecting the policy head with the largest expected reward (OWL (max Q)) and versus selecting the policy with the largest expected reward at each step of the roll-out (max Q $\forall t$). We see that approaches which use the Q-value to select the policy fail, as does Exp Replay. Random policy selection performs well, but the MAB performs significantly better than random policy selection, a t-test with unequal variances has a $p$-value of 0.06 for the SC policies, there isn't a significant difference for the SC+DK policies. Thus there are statistically significant benefits to using the MAB. The MAB approach can also behave similarly to a random policy head selection can perform well (Bergstra and Bengio 2012; Mania, Guy, and Recht 2018).

**Generalization Results.** Many works have focused on generalization properties of RL agents in the MiniGrid environment (Goyal et al. 2020), training on hundreds of levels. Instead, we train on just 5 levels sequentially, which produces a significant risk of overfitting. We take the final SC and SC+DK policies and evaluate them on 100 different, unseen tasks Figure 7. We find that our OWL agent is able to transfer effectively to these unseen tasks, solving up to 40% of unseen levels single walled levels, around 4× more than Exp Replay. OWL can even solve harder environments where Exp Replay totally fails. This exciting result demonstrates that the OWL agent is able to re-use each of the base policies learnt sequentially for solving totally different tasks. Randomly selecting policies in the roll-out is a strong strategy which the MAB can emulate. This demonstrates potential for our approach in a hierarchical RL setting, with links to options (Bacon, Harb, and Precup 2017).

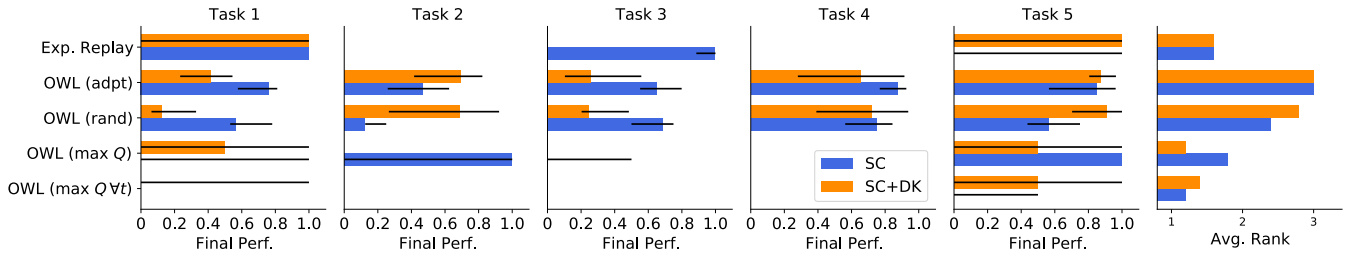


Figure 6: Final performance for different OWL policy selection strategies and Exp Replay.

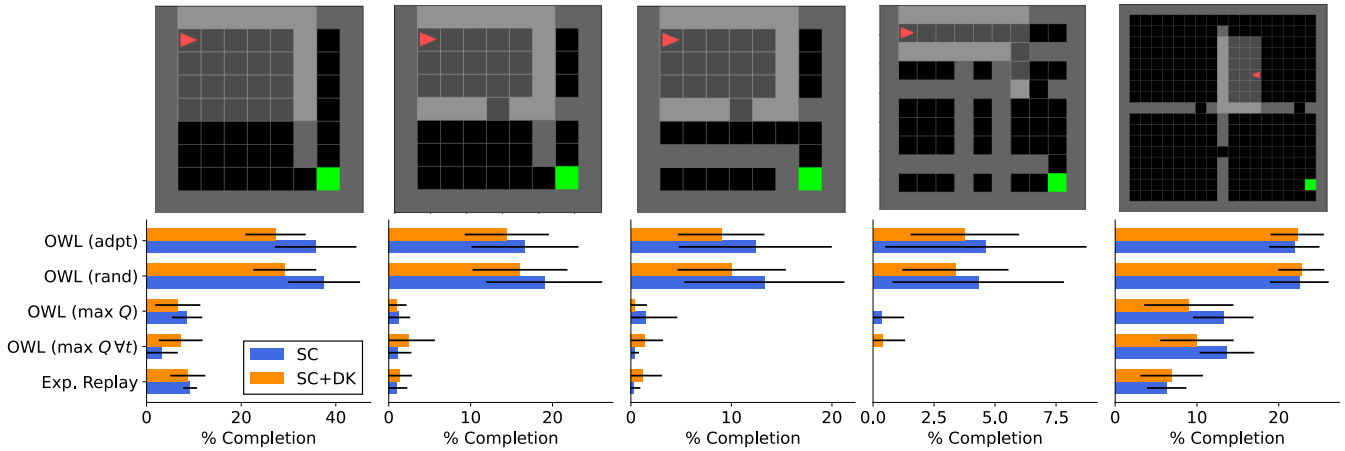


Figure 7: Mean and std error proportion of successes for 100 different environments which have not been seen during training for 10 seeds. OWL is able to generalize to unseen environments while Exp Replay fails.

### 6.3 Ablations

OWL decreases in performance when we remove the EWC, Table 1. By replacing the EWC with a distillation regularization which ensures that the outputs from the previous task's $Q$-function remain similar to the previous task's $Q$-function for the current task (Hinton, Vinyals, and Dean 2015; Li and Hoiem 2017), also decreases performance, see Section D.2 for more details. Distillation regularization works well for classification problems, however we are performing regression, which could explain the drop in performance compared to EWC. We also compare to a Full Rehearsal (FR) which is an upper bound to OWL performance. FR has a buffer for each task and a separate policy head for each task, as such it does not scale gracefully as the number of tasks increase in comparison to OWL, see Section G in the appendix for implementation details.

| | SC | SC+DK |
|---|---|---|
| Exp Replay | 0.01 (0.61, 0.00) | 0.00 (0.52, 0.00) |
| OWL (orcl) | **0.85** (0.97, 0.72) | **0.60** **(0.98, 0.44)** |
| OWL (adpt) | 0.59 (0.75, 0.48) | **0.63** **(0.79, 0.45)** |
| OWL - EWC (orcl) | 0.45 (0.53, 0.39) | 0.40 (0.48, 0.30) |
| OWL - EWC (adpt) | 0.49 (0.60, 0.39) | **0.50** **(0.62, 0.37)** |
| OWL - EWC + DL (orcl) | 0.45 (0.53, 0.36) | 0.34 (0.40, 0.29) |
| OWL - EWC + DL (bndt) | 0.53 (0.61, 0.38) | 0.39 (0.45, 0.33) |
| Full Rehearsal | 0.99 (0.99, 0.97) | 0.99 (1.00, 0.98) |

Table 1: Comparisons and ablations for OWL evaluating on the 5 SC and SC+DK tasks for 10 seeds.

## 7 Conclusion and Future Work

We consider a challenging CRL setting where *different tasks* have the *same observation*. We showed that established experience replay methods which are task agnostic with a single predictor network fail due to interference. Our main contribution is to highlight this interference problem and introduce a simple yet effective approach for this paradigm, which we call OWL. OWL is able to limit forgetting while training on tasks sequentially by using a $Q$-function with a shared feature extractor and a population of linear heads for each task. OWL does not require knowledge of the task at test time, but is still able to achieve close to optimal performance using a multi-armed bandits algorithm. We evaluated OWL on challenging RL environments such as MiniGrid, where we were able to solve five different tasks with similar observations. Finally, we showed it is possible to transfer our learned policies to unseen and more difficult environments.

There are a variety of exciting future directions for this work. For instance, exploring change detection methods using more robust probabilistic models in RL, to detect shifts in reward and state-action distributions.

## References


Ahn, H.; Cha, S.; Lee, D.; and Moon, T. 2019. Uncertainty-based Continual Learning with Adaptive Regularization. In *Neural Information Processing Systems*.

Aljundi, R.; Caccia, L.; Belilovsky, E.; Caccia, M.; Lin, M.; Charlin, L.; and Tuytelaars, T. 2019a. Online continual learning with maximally interfered retrieval. *arXiv preprint arXiv:1908.04742*.

Aljundi, R.; Lin, M.; Goujaud, B.; and Bengio, Y. 2019b. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*.

Bacon, P.-L.; Harb, J.; and Precup, D. 2017. The Option-Critic Architecture. In *AAAI*, 1726–1734.

Balaji, Y.; Farajtabar, M.; Yin, D.; Mott, A.; and Li, A. 2020. The Effectiveness of Memory Replay in Large Scale Continual Learning. *arXiv preprint arXiv:2010.02418*.

Ball, P.; Parker-Holder, J.; Pacchiano, A.; Choromanski, K.; and Roberts, S. 2020. Ready Policy One: World Building Through Active Learning. *International Conference on Machine Learning*.

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; Van Hasselt, H.; and Silver, D. 2016. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*.

Bellemare, M.; Candido, S.; Castro, P.; Gong, J.; Machado, M.; Moitra, S.; Ponda, S.; and Wang, Z. 2020. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588: 77–82.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2012. The Arcade Learning Environment: An Evaluation Platform for General Agents. *CoRR*, abs/1207.4708.

Bellman, R. 1957. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5): 679–684.

Benjamin, A. S.; Rolnick, D.; and Kording, K. P. 2019. Measuring and Regularizing Networks in Function Space. In *International Conference on Learning Representations*.

Bergstra, J.; and Bengio, Y. 2012. Random Search for Hyper-Parameter Optimization. In *Journal of Machine Learning Research*.

Bishop, C. M. 2006. *Pattern recognition and machine learning*. springer.

Bishop, C. M.; and Svensén, M. 2012. Bayesian hierarchical mixtures of experts. *arXiv preprint arXiv:1212.2447*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. arXiv:1606.01540.

Caccia, M.; Rodriguez, P.; Ostapenko, O.; Normandin, F.; Lin, M.; Page-Caccia, L.; Laradji, I. H.; Rish, I.; Lacoste, A.; Vázquez, D.; et al. 2020. Online fast adaptation and knowledge accumulation (osaka): a new approach to continual learning. *Advances in Neural Information Processing Systems*, 33.

Cesa-Bianchi, N.; and Lugosi, G. 2006. *Prediction, learning, and games*. Cambridge university press.

Chaudhry, A.; Facebook, M. R.; Research, A. I.; Elhoseiny, M.; Ajanthan, T.; Dokania, P. K.; Torr, P. H. S.; and Ranzato, M. . A. 2019. On Tiny Episodic Memories in Continual Learning. *arxiv.org:1902.10486*.

Chevalier-Boisvert, M.; Willems, L.; and Pal, S. 2018. Minimalistic Gridworld Environment for OpenAI Gym. https://github.com/maximecb/gym-minigrid.

Chua, K.; Calandra, R.; McAllister, R.; and Levine, S. 2018. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Advances in Neural Information Processing Systems 31*, 4754–4765.

Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; et al. 2018. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning*.

Eysenbach, B.; and Levine, S. 2019. If MaxEnt RL is the Answer, What is the Question? arXiv:1910.01913.

Farquhar, S.; and Gal, Y. 2018. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*.

Goyal, A.; Sodhani, S.; Binas, J.; Peng, X. B.; Levine, S.; and Bengio, Y. 2020. Reinforcement Learning with Competitive Ensembles of Information-Constrained Primitives. In *International Conference on Learning Representations*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018a. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, 1861–1870.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; and Levine, S. 2018b. Soft Actor-Critic Algorithms and Applications. *CoRR*, abs/1812.05905.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep Reinforcement Learning that Matters. *AAAI*.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Deepmind, S. 2017. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *AAAI*. ISBN 1710.02298v1.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Isele, D.; and Cosgun, A. 2018. Selective Experience Replay for Lifelong Learning. In *AAAI*.

Khetarpal, K.; Riemer, M.; Rish, I.; and Precup, D. 2020. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*.

Kingma, D. P.; and Lei Ba, J. 2015. ADAM: A Method for Stochastic Optimization. In *ICLR*.

Kingma, D. P.; and Welling, M. 2013. Auto-Encoding Variational Bayes. *CoRR*, abs/1312.6114.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N. C.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Neural Information Processing Systems*.

Lee, S.; Ha, J.; Zhang, D.; and Kim, G. 2020. A Neural Dirichlet Process Misture Model for Task-Free Continaul Learning. In *International Conference on Learning Representations*.

Li, Z.; and Hoiem, D. 2017. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Lin, L.-J. 1992. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Mach. Learn.*, 8(3–4): 293–321.

Lin, X.; Zhen, H.-L.; Li, Z.; Zhang, Q.; and Kwong, S. 2019. Pareto multi-task learning. *arXiv preprint arXiv:1912.12854*.

Lopez-Paz, D.; and Ranzato, M. . A. 2017. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*.

Mania, H.; Guy, A.; and Recht, B. 2018. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*.

Mendez, J. A.; Wang, B.; and Eaton, E. 2020. Lifelong Policy Gradient Learning of Factored Policies for Faster Training Without Forgetting. In *Advances in Neural Information Processing Systems*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*.

Nekoei, H.; Badrinaaraayanan, A.; Courville, A.; and Chandar, S. 2021. Continuous Coordination As a Realistic Scenario for Lifelong Learning. arXiv:2103.03216.

Nguyen, C. V.; Li, Y.; Bui, T. D.; and Turner, R. E. 2018. Variational Continual Learning. In *International Conference on Learning Representations*.

Nix, D. A.; and Weigend, A. S. 1994. Estimating the mean and variance of the target probability distribution. In *IEEE International Conference on Neural Networks - Conference Proceedings*, volume 1, 55–60.

OpenAI; Andrychowicz, M.; Baker, B.; Chociej, M.; Józefowicz, R.; McGrew, B.; Pachocki, J. W.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; Schneider, J.; Sidor, S.; Tobin, J.; Welinder, P.; Weng, L.; and Zaremba, W. 2018. Learning Dexterous In-Hand Manipulation. *CoRR*, abs/1808.00177.

Rajendran, J.; Irpan, A.; and Jang, E. 2020. Meta-Learning Requires Meta-Augmentation. In *Advances in Neural Information Processing Systems 33*.

Ring, M. B. 1994. *Continual learning in reinforcement environments*. Ph.D. thesis, University of Texas at Austin.

Robert M. French. 1999. Catastrophic forgetting in connectionists networks. *Trends in Cognitive Sciences*, 3(4): 128–135.

Rolnick, D.; Ahuja, A.; Schwarz, J.; Lillicrap, T.; and Wayne, G. 2019. Experience Replay for Continual Learning. In *Advances in Neural Information Processing Systems 32*, 350–360.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive Neural Networks. *CoRR*, abs/1606.04671.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*.

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

Schwarz, J.; Czarnecki, W.; Luketina, J.; Grabska-Barwinska, A.; Teh, Y. W.; Pascanu, R.; and Hadsell, R. 2018. Progress & Compress: A scalable framework for continual learning. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 4535–4544. PMLR.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nat.*, 529(7587): 484–489.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

van de Ven, G. M.; and Tolias, A. S. 2018. Three scenarios for continual learning. In *NeurIPS Continual Learning workshop*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2015. Deep Reinforcement Learning with Double Q-learning. In *AAAI*. ISBN 1509.06461v3.

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

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

Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; Finn, C.; University, S.; Berkeley, U. C.; and At Google, R. 2020. Gradient Surgery for Multi-Task Learning. In *Advances in Neural Information Processing Systems*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*.