UNCLEAR: A Straightforward Method for Continual Reinforcement Learning

Samuel Kessler [1] **Jack Parker-Holder** [1] **Philip Ball** [1] **Stefan Zohren** [1] **Stephen J. Roberts** [1]

Abstract

We present UNCLEAR, a new approach for continual reinforcement learning in the face of tasks which catastrophically interfere. At the heart of UNCLEAR is a value function, which is parameterized by a neural network feature extractor *shared across all tasks*, and a set of linear heads, each *specializing on a single task*. We are then able to train one policy with respect to each head. Forgetting is prevented using simple regularization of the feature extraction layers of the policy and Q-functions. Drawing inspiration from online learning, we introduce a novel means to select policies at test time, allowing us to automatically select the right policy for an unknown task. We show in a simple experiment that our approach is able to achieve close to optimal performance when training sequentially on orthogonal tasks.

1. Introduction

Reinforcement Learning (RL, (Sutton 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), leading to excitement that RL may soon deliver real-world autonomous agents. 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., 2017)).

By contrast, Continual learning (CL) is a paradigm whereby an agent sequentially learns new tasks. The agent loses access to the training data from previous tasks and is continuously evaluated on these previous tasks, with the aim of preventing *catastrophic forgetting* of previous tasks. Forgetting is the consequence of overwriting weights which were important for previous tasks while learning a new task. This is challenging, since training data from the previous tasks is generally lost. There has recently been significant progress made, through a variety of regularization techniques (Kirkpatrick et al., 2016; Nguyen et al., 2018; Benjamin et al., 2019), rehearsal and reply methods (Lopez-Paz and Ranzato, 2017; Shin et al., 2017) or by mixture of expert networks (Rusu et al., 2016; Lee et al., 2020).

Recent advances at the intersection of these two fields, *Continual Reinforcement Learning* (CRL), have shown RL agents successfully learn RL tasks sequentially (Schwarz et al., 2018; Rolnick et al., 2019). However, these methods typically lack a mechanism to detect different tasks. This becomes necessary when different tasks *interfere*: that is tasks are fundamentally orthogonal and an agent trained on both tasks significantly underperforms separate agents trained on these individual tasks. This issue prevents the field from realizing the goal of autonomous agents that can sequentially learn new tasks in the real-world. Interfering tasks can arise naturally in RL as we demonstrate later.

Our main contribution is a new approach for CRL, which we call UNcertainty guided Continual LEARning, or UNCLEAR. UNCLEAR uses an off-policy RL algorithm to train a state-value function across all tasks sequentially. The key insight is we can use a single network with multiple heads, parameterized by linear layers, to fit individual tasks. We then train a separate policy with respect to each head. At test time, UNCLEAR adaptively selects an appropriate policy using a method inspired by online learning. We evaluate UNCLEAR on a set of orthogonal tasks, and despite training sequentially it achieves close to the optimal performance.

2. Related Work

Continual Learning in Supervised Learning. Continual Learning (CL) can be viewed as a sequential learning problem. One approach to learning in this setting is through online Bayesian inference, which can be interpreted as a weight space regularization. A popular approach is Elastic Weight Consolidation (EWC) which performs makes use of a diagonal Laplace approximation (Kirkpatrick et al., 2016). Variational Inference has also been used successfully (Nguyen et al., 2018; Kessler et al., 2019). Other methods which regularize neural network weights have also found success (Zenke et al., 2017; Chaudhry et al., 2018).

*Equal contribution [1]University of Oxford. Correspondence to: Samuel Kessler <skessler@robots.ox.ac.uk>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

By contrast, expansion approaches to CL work by adding new neural resources to enable learning new tasks while preserving components for specific tasks (Rusu et al., 2016). Rehearsal and replay methods have been found to be very effective and involve replaying data from previous tasks with a generative model (Shin et al., 2017) or a replay buffer (Lopez-Paz and Ranzato, 2017; Aljundi et al., 2019).

Continual Learning in Reinforcement Learning. EWC applied to DQN (Mnih et al., 2015) has been used for learning over a series of Atari tasks. The task changes during training are inferred using a Hidden Markov Model (HMM) from pixels (although it is not clear whether the task is inferred for evaluation). (Kirkpatrick et al., 2016). Both Progressive Networks (PN, (Rusu et al., 2016)) and Progress and Compress (P&C, (Schwarz et al., 2018)) are applied to policy and value function feature extractors for an actor-critic approach (Sutton et al., 1998). Task boundaries are shown when training different tasks. By contrast, our proposed method is able to infer the task during evaluation.

Leveraging experience replay buffers (Lin, 1992), CLEAR uses actor-critic with V-trace importance sampling (Espeholt et al., 2018) of past experiences from the replay buffer to prevent catastrophic forgetting (Rolnick et al., 2019). On-policy updates are plastic but past experiences (off-policy) are replayed.

Beyond Single Task RL It is also worth noting there has been tremendous excitement with regard to RL algorithms which can generalise across multiple environments. This has either been from scratch (Badia et al., 2020; Silver et al., 2017; Schrittwieser et al., 2019), or by training across an entire task distribution and adapting quickly to a new task (Finn et al., 2017). These differ in that they do not deal with catastrophic forgetting in the same way as CL methods.

Compact RL Policies: We also draw inspiration from recent work demonstrating the efficacy of sparse neural networks (Frankle and Carbin, 2019). There have been approaches utilizing this idea in RL (Mania et al., 2018; Cuccu et al., 2019; Ha and Schmidhuber, 2018; Choromanski et al., 2018). Our work differs in two ways: (1) we learn *multiple* linear policies, representing individual skills/task and (2) our linear policies include both mean and variance estimates.

3. Background

3.1. Reinforcement Learning

A Markov Decision Process (MDP, (Bellman, 1957)) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. 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 $a_t \in \mathcal{A}$. $\mathcal{P}(s_{t+1}|s_t, a_t)$ is the probability that the system/agent transitions from s_t to s_{t+1} given action a_t and $\mathcal{R}(a_t, s_t, s_{t+1})$ is a reward obtained by an agent transitioning from s_t to s_{t+1} via a_t. A policy $\pi_\phi : \mathcal{S} \to \mathcal{A}$ is a mapping from $\mathcal{S}$ to $\mathcal{A}$, parameterized by ϕ. In this paper we consider MDPs with finite horizons H. The return from a state is defined as the sum of discounted future reward $R_t = \sum_{i=t}^{H} \gamma^{(i-t)} r(s_i, a_i)$, for some discount factor $\gamma \in (0, 1)$. Thus, the RL objective is to maximize $\mathcal{L} = \mathbb{E}_{a_i \sim \pi}[R_1 | s_0]$ given an initial state s_0, sampled from the environment.

It is common practice to learn an action-value for each state s_t, as $Q^\pi(s_t, a_t) = \mathbb{E}_{s_t \sim \mathcal{P}, r_t \sim \mathcal{R}, a_t \sim \pi}[\sum_t R_t]$. We parameterize the action-value function with a neural network, denoted $Q(s_t, a_t; \theta_t)$, with parameters θ_t and seek to update θ_t as follows:

$$\theta_{t+1} \leftarrow \theta_t + \eta(y_t^Q - Q(s_t, a_t; \theta_t))\nabla_\theta Q(s_t, a_t; \theta_t)$$

where η is the scalar learning rate and $y_t^Q = r_t + \gamma \max_a Q(s_{t+1}, a; \theta^-)$ is the target value. The parameters θ^- are target network parameters which are periodically updated $\theta^- = \theta_t$. Typically, this update is done with samples from a replay buffer (Lin, 1992).

We train the policy π_ϕ to maximize the Q-values. In this paper we use the Soft Actor Critic (SAC, (Haarnoja et al., 2018a;b)) algorithm, thus include an entropy term, adopting a Q-function update of the form:

$$y_t^Q \leftarrow r_t + \gamma(Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - \alpha \log(\pi_\phi(a_{t+1}|s_{t+1}))).$$

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 given a set of M tasks sequentially $\mathcal{T}_i$ for $i = 1, \dots M$. Where each task $\mathcal{T}_i$ is comprised of a tuple $(p_i(x), p_i(y|x))$ and hence a dataset with input $X \in \mathbb{R}^d$ and output $y \in \mathbb{R}$. In the context of RL for a policy $x := s_t$ and $y := a_t$ and for the action-value function $x := (s_t, a_t)$ and $y := y_t^Q$. Although the model will lose access to the training dataset for task $\mathcal{T}_i$, it will be continually evaluated on all previous tasks $\mathcal{T}_i$ for $i \leq t$. For a comprehensive review of CL scenarios see (van de Ven and Tolias, 2018).

A common method to alleviating forgetting in NNs is to use EWC which constrains the learning of a new parameters anchoring at the previous task's optimal parameters with an L^2 regularization. EWC weights the regularization by parameter importance by using the diagonal empirical Fisher information (FI). After performing task $\mathcal{T}_i$, the regularization for the next task $\mathcal{T}_{i+1}$ is:

$$\mathcal{L}_{\text{EWC}}^{(i)}(\theta) = \frac{\lambda}{2} F_i \left\| \theta - \theta_i^* \right\|^2, \tag{1}$$

where F_i is the diagonal empirical FI for task for task i. The hyperparameter λ enables us to increase or decrease the size of the overall regularization. The FI corresponds to the precision of a Laplace approximation hence the larger the FI, the smaller the parameter variance and thus the larger the regularization and vise-versa. Since EWC leads to a linear increase in memory (w.r.t i), methods such as Online-EWC (Schwarz et al., 2018) can be used. Additional details are in the Appendix (Sec. B).

3.3. Online Learning

Online Learning is a class of methods designed to tackle sequential decision making processes. In particular, a learner takes actions $a \in \mathcal{A}$, whilst learning from a sequence of data $u_1, u_2, \ldots, u_T$ that arrive in incremental rounds n. The learner must take an action at the beginning of each round, and receives a loss $\ell(a_t, u_t)$. The *regret* is the difference between the optimal choice, $\min_{a \in \mathcal{A}} \ell(a, u_t)$ and the action taken $\ell(a_t, u_t)$. The goal is to minimize the cumulative regret, i.e. the sum of this difference over all future timesteps. At each round t the learner only receives feedback for its chosen action a_t and doesn't see what the feedback would have been for a different action. Based on this feedback the learner updates how it selects future actions.

4. UNCLEAR

At a high level, UNCLEAR uses an off-policy RL algorithm, Soft Actor-Critic (SAC, (Haarnoja et al., 2018a;b)), to train a Q-function across tasks sequentially. The key insight is that we can use a single network with shared feature representation layers, regularized with EWC updates. We then train multiple heads, parameterized by linear layers, to fit individual tasks. We train a separate policy with respect to each Q-function head. At test time, we adaptively select the head using a method inspired by online learning. In this section we provide additional details on each component.

Common CL tasks involve $(p_i(x), p_i(y|x))$ changing simultaneously. For instance the common benchmark Split MNIST involves a series 5 of binary classification tasks from MNIST, each digit has a different distribution over pixels, $p_i(x)$ changes for each task. In this paper we ask a different question: what if $p_i(x)$ stays the same and the reward distribution changes then the conditional target y_t^Q and action a_t distributions, $p_i(y|x)$, change for each task i. In RL this is easy to construct by simply changing the reward function, as we do in our experiments (see: Sec: 5)

Alleviating forgetting. In the scenarios which we consider, we have the same state-action space across tasks but the reward function changing. We construct our policy such that $\pi_\phi : s_t \to z_t^\pi \to a_t$, mapping to representation z_t^π which is shared across tasks. The parameters $\phi = \{\phi_z, \phi_{1:M}\}$ where ϕ_z are the neural network feature extraction layers, and $\phi_{1:M}$ are linear policies. Similarly for the action-value function $Q_\theta : (s_t, a_t) \to z_t^Q \to y_t^Q$, we want to learn a feature representation function that is shared across tasks the parameters $\theta = \{\theta_z, \theta_{1:M}\}$ where θ_z is the neural network feature extraction layers, and $\theta_{1:M}$ are linear heads.

Multi-head architectures are commonly used in CL. Instead of multiple networks, we train one network across all tasks, but use multiple heads, each parameterized by a linear layer. We use a probabilistic neural network (Nix and Weigend, 1994), which has been shown to be effective as an ensemble in model-based RL (Chua et al., 2018; Janner et al., 2019). Each head models a *separate* task allowing us to learn from tasks which might interfere. We also address forgetting in the shared neural network feature extractors for the action-value function and policy across tasks using EWC (Kirkpatrick et al., 2016). We can proceed to train our agent according to Algorithm 1.

Algorithm 1 UNCLEAR: Training

Input: Tasks $\mathcal{T} = \{\mathcal{T}_i\}_{i=1}^M$, regularizations $\Omega^Q = \emptyset, \Omega^\pi = \emptyset$
Initialize: models.
for $\mathcal{T}_i \in \mathcal{T}$ **do**
1. Train Q-function with parameters $\{\theta_z, \theta_i\}$ and policy with parameters $\{\phi_z, \phi_i\}$ using SAC with regularizations Ω^Q and Ω^π.
2. Calculate Q-function EWC regularization in Equation 1 and $\Omega^Q := \{\mathcal{L}^Q_{\text{EWC}}, \Omega^Q\}$.
3. Calculate policy regularization in Equation 1 and $\Omega^\pi := \{\mathcal{L}^\pi_{\text{EWC}}, \Omega^\pi\}$.
4. Empty the experience reply buffer $\mathcal{D} = \emptyset$.

Adaptive Task Inference. Now we have a means to train multiple policies, we must adaptively select the best policy for each test task. We return to the description of online learning, in Section 3.3. In our approach we consider the set of actions $\mathcal{A}$ to be the policy 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 take inspiration from (Ball et al., 2020) and use a modified version of the Exponentially Weighted Average Forecaster algorithm (Cesa-Bianchi and Lugosi, 2006). 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 updating ℓ_t as follows:

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

For some step size parameter η. We consider the case where the selection of ϕ_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 $l_{i_t} = \hat{G}_{\phi_t}(\theta_{i_t})$, where $G_{\phi_t}(\theta_{i_t})$ is 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 (see Appendix D for details), hence $\hat{G}$. Henceforth we denote by $\mathbf{p}_\phi^t$ the exponential weights distribution over ϕ values at time t. The pseudocode for our test-time procedure is shown in Algorithm 2. For more details see the Appendix (Sec: D.3).

Algorithm 2 UNCLEAR: Testing

Input: unknown test task $\mathcal{T}_j$, policies $\{\phi_i\}_{i=1}^M$, step size η, number of timesteps T.
Initialize: $\mathbf{p}_\phi^1$ as a uniform distribution, s_1 as the initial state of the test task.
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}
3. Use Equation 2 to update $\mathbf{p}_\phi^t$ with

$$l_{i_t}^t = \hat{G}_{\phi_t}(\theta_{t+1})$$

5. Experiments

We evaluate UNCLEAR on a simple, yet challenging problem, based on the Pendulum-v0 environment from the OpenAI Gym (Brockman et al., 2016). Typically, the policy is rewarded for placing the pendulum at $0°$. Instead, we amend the reward function to produce two orthogonal tasks, with optimal positions: $\{+90°, -90°\}$. We train on each task three times, switching every $20,000$ timesteps. We compare the following methods:

- UNCLEAR (Oracle): UNCLEAR with knowledge of the task index at test time.
- UNCLEAR (Adaptive): UNCLEAR with adaptive test-time task inference.
- SAC (Sequential): A baseline SAC agent which trains on both tasks sequentially. SAC's experience replay buffer enables it to mimic CLEAR.
- SAC (Single): Two separate SAC agents which train on the individual tasks. We record the final performance after $120,000$ timesteps to indicate the maximum performance available to all agents.

Fig. 1 shows the main results, which were run for a total of 10 seeds. Importantly, we see that the Oracle version of UNCLEAR (orange), which knows the test task index, is able to perform as well as two separate SAC policies trained on the individual tasks (black, dashed). What is most encouraging however, is that we can almost achieve this same performance without informing the agent of the task index, using our adaptive mechanism (blue).

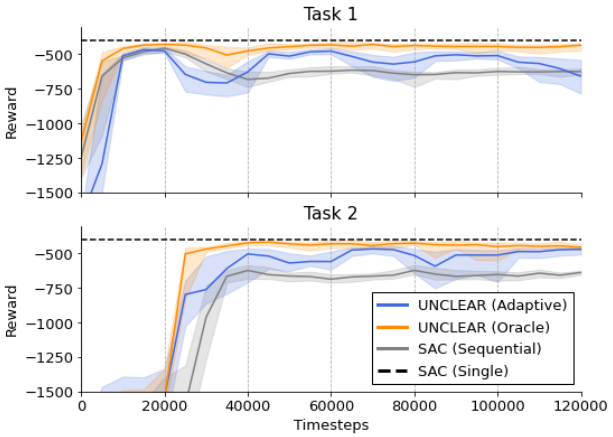

Figure 1. Median performance across 10 seeds. Shaded regions correspond to the Inter-Quartile Range.

In addition, the orthogonality of the tasks leads to catastrophic interference: the sequential SAC agent fails to learn either task well (grey). Instead, it places the pendulum at $0°$, in the middle of the two goals, achieving a suboptimal reward on both tasks. Note that each time we switch tasks, the UNCLEAR agent flushes the replay buffer. We believe smarter use of this could also increase performance of CLEAR. Additional results exploring the importance of the probabilistic networks are in the Appendix, Sec: C.

6. Conclusion and Future Work

We introduced UNCLEAR, a simple yet effective approach for continual reinforcement learning. UNCLEAR is able to limit forgetting while training on sequential tasks, using an action-value function with a shared feature extractor and an ensemble of linear heads. Crucially, UNCLEAR does not require knowledge of the task index at test time, but is still able to achieve close to optimal performance.

There are a variety of exciting future directions for this work. Notably, it may also be possible to switch from weight to functional regularization on the policies. This has been shown to be effective for policy gradient methods (Schulman et al., 2015; 2017). Also it would be more natural to detect task boundaries during training in addition to evaluation. The notion of uncertainties for detecting changes in the domain has previously been show (Titsias et al., 2020) however enabling detection of changing reward distributions or both is an interesting possible future direction.

It is also possible our method may work in a single RL task. Rather than consider task boundaries as entire environments, we could consider continual learning options (Bacon et al., 2017), and switch between options using an adaptive mechanism. This has been shown to be effective for example in environments with multiple distinct regions, which may cause catastrophic interference (Fedus et al., 2020).

References

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019). Gradient based sample selection for online continual learning. In *Neurips*.

Bacon, P.-L., Harb, J., and Precup, D. (2017). The option-critic architecture. In *AAAI*, pages 1726–1734.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., and Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. *CoRR*, abs/2003.13350.

Ball, P., Parker-Holder, J., Pacchiano, A., Choromanski, K., and Roberts, S. (2020). Ready policy one: World building through active learning. *accepted to ICML 2020*.

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*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym.

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

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. S. (2018). Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In *ECCV*.

Choromanski, K., Rowland, M., Sindhwani, V., Turner, R., and Weller, A. (2018). Structured evolution with compact architectures for scalable policy optimization. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 970–978, Stockholmsmässan, Stockholm Sweden. PMLR.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 4754–4765.

Cuccu, G., Togelius, J., and Cudré-Mauroux, P. (2019). Playing atari with six neurons. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 998–1006, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

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. *arXiv preprint arXiv:1802.01561*.

Eysenbach, B. and Levine, S. (2019). If maxent rl is the answer, what is the question?

Fedus, W., Ghosh, D., Martin, J. D., Bellemare, M. G., Bengio, Y., and Larochelle, H. (2020). On catastrophic interference in atari 2600 games. *CoRR*, abs/2002.12499.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia. PMLR.

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Ha, D. and Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NeurIPS'18, pages 2455–2467, USA. Curran Associates Inc.

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 Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmässan, Stockholm Sweden.

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. (2017). Deep reinforcement learning that matters. *CoRR*, abs/1709.06560.

Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: Model-based policy optimization. *CoRR*, abs/1906.08253.

Kessler, S., Nguyen, V., Zohren, S., and Roberts, S. (2019). Indian Buffet Neural Networks for Continual Learning. In *4th workshop on Bayesian Deep Learning (NeurIPS 2019)*.

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 Deepmind, C. B. (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*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8(3–4):293–321.

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

Mania, H., Guy, A., and Recht, B. (2018). Simple random search of static linear policies is competitive for reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 1800–1809. Curran Associates, Inc.

Martens, J. (2014). New insights and perspectives on the natural gradient method. Technical report.

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*.

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, pages 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.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. (2019). Experience replay for continual learning. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 350–360. Curran Associates, Inc.

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.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T. P., and Silver, D. (2019). Mastering atari, go, chess and shogi by planning with a learned model. *CoRR*, abs/1911.08265.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning (ICML)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

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., editors, *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*, pages 4535–4544. PMLR.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual Learning with Deep Generative Replay. 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.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

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

Titsias, M. K., Schwarz, J., de G. Matthews, A. G., Pascanu, R., and Teh, Y. W. (2020). Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*.

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

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

Appendicies

A. Additional Definitions

B. Elastic Weight Consolidation

A common method to alleviating forgetting in NNs is to use EWC which constrains the learning of a new parameters by using anchoring at the previous task's optimum with an L^2 regularisation. Inspired by sequential Bayesian updates using a Laplace approximate posterior around the Maximum a Posteriori (MAP) solution: $q(\theta|\theta^*, \bar{H}^{-1})$. The expected Hessian of the log posterior, $\bar{H} = \mathbb{E}[H]$ can be approximated by a sum of outer products of Jacobians, as long as the NN fits the data well and the residuals are small, this is referred to as the Gaussian-Newton matrix (GGN). The GGN matrix corresponds Fisher information (FI) for exponential family distributions (Martens, 2014). EWC weights the regularisation parameter importance by using the diagonal empirical FI. The EWC regularisation for task i is:

$$\mathcal{L}(\theta) = \sum_{k=1}^{i-1} \frac{\lambda}{2} F_k \left\| \theta - \theta_k^* \right\|^2 \tag{3}$$

where F_k is the diagonal empirical FI for task for previous task k for all $k < i$. λ is a hyperparameter which enables us to increase or decrease the size of the parameter regularisation. The FI corresponds to the precision hence the larger the FI, the smaller the uncertainty of that parameter and thus the larger the regularisation and vise-versa.

C. Ablation Studies

Do we need uncertainty aware models? In this section we consider the impact of our model design. As discussed, the feedback for our online learning mechanism incorporates both the mean and variance predictions, thus considering aleatoric uncertainty. To assess the impact of this, we also ran UNCLEAR with a simple MSE feedback, ignoring the variance head. In Fig. 2 we show both the performance of this approach, as well as the effectiveness of the online learning mechanism.

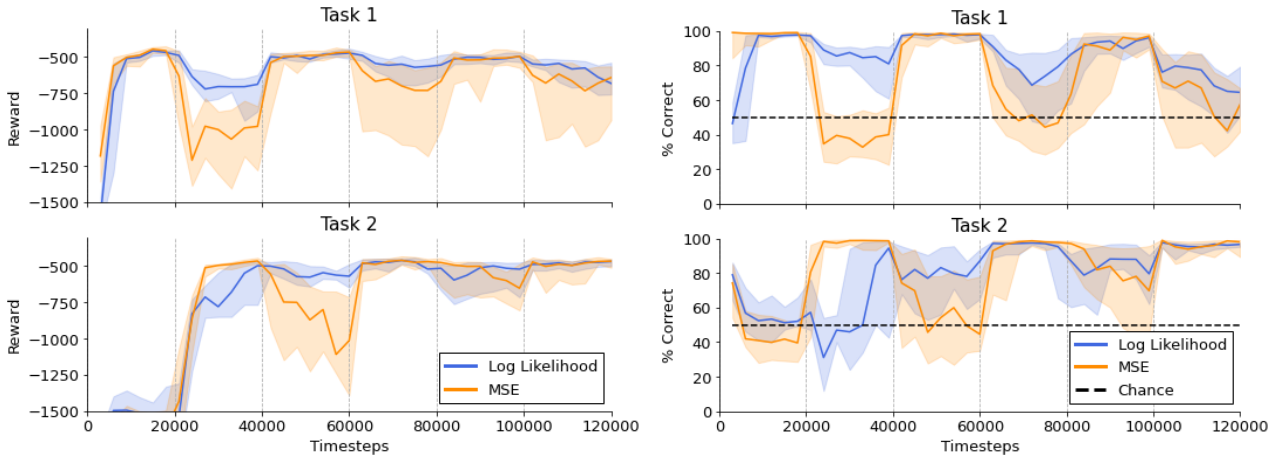

Figure 2. Left: Median performance across 10 seeds. Shaded regions correspond to the Inter-Quartile Range. Right: The performance of the online learning mechanism for both tasks. Curves are the median percentage correct, shaded regions correspond to the Inter-Quartile Range.

It is clear to see here that the negative log-likelihood is significantly more effective. The linkage between selecting the correct policy and final performance is evident, and justifies our use of the probabilistic models.

D. Implementation Details

D.1. Soft Actor-Critic

We implement Soft Actor-Critic (SAC) in PyTorch, following the learned reward temperature approach of (Haarnoja et al., 2018b). Briefly, Soft Actor-Critic aims to maximise the sum of the reward and the entropy of the policy over the task

horizon; this results in behaviour that can be summarised as "maximising reward while acting as randomly as possible", and can be shown as being optimal in a meta-POMDP setting (Eysenbach and Levine, 2019). What is difficult about such a dual objective is that ultimate task performance is sensitive to the trade-off between randomness/entropy and reward (Haarnoja et al., 2018a). In the continual setting that we present here, different tasks over the agent's lifetime may require different degrees of reward/entropy scaling, which would require specific tuning of this parameter (usually denoted as α) per new task. It is possible however to learn this parameter α from the task by introducing a 'target entropy' ($\hat{\mathcal{H}}$), which actively scales reward against policy entropy to achieve a target entropy of $\bar{\mathcal{H}} = -\dim(\mathcal{A})$, where $\mathcal{A}$ is the action space of an environment. This results in α that adapts to its current policy and experiences it receives from the environment.

The Q-function used for UNCLEAR is identical to the architecture from PETS which has been shown to be successful in Model-Based RL (Chua et al., 2018). UNCLEAR uses a multi-head architecture where the number of heads equals the number of tasks. Each head has a mean and variance mapping (Nix and Weigend, 1994). SAC uses Q-functions with four layers with Relu activations. An ensemble of two Q-functions are used for compensation for an optimistic Q-value as is standard is SAC. The target Q-function is update using a exponentially weighted moving average with $\gamma = 0.99$.

The Gaussian policy is parameterised by a NN which uses mean and variance heads and share three NN layers with Relu activations, it is trained with a learning rate of 3×10^{-4}. Gradient based learning is performed using the Gaussian reparameterisation trick (Kingma and Welling, 2013).

D.2. Elastic Weight Consolidation

The EWC implementation for both the Q-function and policy uses $\lambda = 100$. The weighting of the L^2 regularisation is performed with the empirical Fisher Information $F(\theta) = \sum_n \nabla_\theta \log p_\theta(y_n|x_n) \nabla_\theta \log p_\theta(y_n|x_n)^\top$ which is calculated using all samples from the experience replay buffer upto a maximum of $60,000$ samples. EWC is applied on the feature extraction layers of the Q-function and policy and to the task specific mean and variance heads for when they are trained upon again. No EWC regularization is directly applied to the target Q-function.

D.3. Online Learning

Computing Feedback Two Q-values are estimated by SAC and UNCLEAR. In UNCLEAR the NN outputs a mean and variance, these estimates can be combined by considering Q-values as a mixture of Gaussians. The Gaussian mixture of L action-value functions has mean and variance $\mu_* = \frac{1}{L}\sum_{l=1}^{L} \mu_l(x)$ and $\sigma_*^2 = \sum_{l=1}^{L}(\sigma_l^2(x) - \mu_l^2(x)) - \mu_*^2(x)$ (Lakshminarayanan et al., 2017). These estimates are then used as the basis for MSE and negative log likelihood feedback to the Exponentially Weighted Average Forecaster algorithm (ExpWeights).

Hyperparameters The losses supplied to ExpWeights are sometimes very large, especially for a feedback in the form of a MSE. Hence we set a threshold to $l_{i_t} = \min(50, l_{i_t})$ where l is a loss, those considered are inverse negative log likelihood and inverse MSE, i_t denotes the index of the policy chosen at time t. The step size is set to $\eta = 0.98$.