Lifelong Learning using Eigentasks: Task Separation, Skill Acquisition, and Selective Transfer

Aswin Raghavan [1] **Jesse Hostetler** [1] **Indranil Sur** [1] **Abrar Rahman** [1] **Ajay Divakaran** [1]

Abstract

We introduce the eigentask framework for lifelong learning. An eigentask is a pairing of a skill that solves a set of related tasks, paired with a generative model that can sample from the skill's input space. The framework extends generative replay approaches, which have mainly been used to avoid catastrophic forgetting, to also address other lifelong learning goals such as forward knowledge transfer. We propose a wake-sleep cycle of alternating task learning and knowledge consolidation for learning in our framework, and instantiate it for lifelong supervised learning and lifelong RL. We achieve improved performance over the state-of-the-art in supervised continual learning, and show evidence of forward knowledge transfer in a lifelong RL application in the game Starcraft 2.

1. Introduction

The goal of lifelong learning (Chen & Liu, 2016; Silver et al., 2013) is to continuously learn a stream of machine learning tasks over a long lifetime, accumulating knowledge and leveraging it to learn novel tasks faster (positive forward transfer) (Fei et al., 2016) without forgetting the solutions to previous tasks (negative backward transfer). The learning problem is non-stationary and open-ended. The learner does not know the number or distribution of tasks, it may not know the identity of tasks or when the task changes, and it must be scalable to accommodate an ever-increasing body of knowledge within a finite model. These characteristics make lifelong learning particularly challenging for deep learning approaches, which are vulnerable to catastrophic forgetting in the presence of non-stationary data.

Many approaches to continual learning (Parisi et al., 2019; Nguyen et al., 2018) and lifelong learning have been studied and applied to computer vision (Hayes et al., 2018; Liu et al., 2020) and reinforcement learning (RL) (Ammar et al., 2015; Tessler et al., 2016) among other areas. Recent work comprises many complementary approaches including learning a regularized parametric representation (Kirkpatrick et al., 2016; 2018), transfer of learned representation (Lee et al., 2019), meta-learning (Nagabandi et al., 2019), neuromodulation (Masse et al., 2018), dynamic neural network architectures (Li et al., 2019; Rusu et al., 2016), and knowledge consolidation using memory (van de Ven & Tolias, 2019).

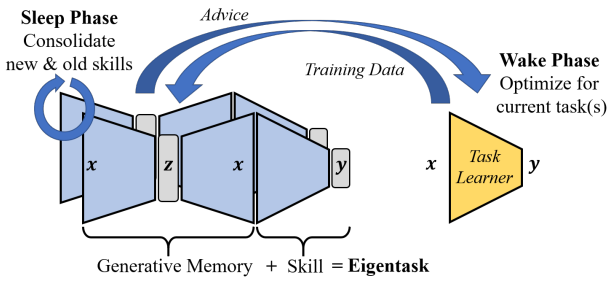


Figure 1. Our approach is based on *eigentasks*, which combine a skill with a generative model of its inputs. Our lifelong learning agents operate in a wake-sleep cycle, solving a stream of tasks during the wake phase, and consolidating new task knowledge during the sleep phase using generative replay.

This paper advances the state of the art in the generative replay approach to lifelong learning (van de Ven & Tolias, 2019; Shin et al., 2017). In this approach, a generative model of the data distribution $p(x, y) = p(x)p(y|x)$ is learned and used for data augmentation i.e., replay data from old tasks when learning a new task. We refer to $p(x)$ as the generator (e.g., of images) and $p(y|x)$ as a skill (e.g., a classifier). Prior work has focused on generative replay as a way to mitigate catastrophic forgetting, i.e., avoiding negative backward transfer. However, forward transfer – the ability to leverage knowledge to quickly adapt to a novel but related task – is an equally important lifelong learning problem that generative replay has not addressed. A structured form of replay is one of the functions of mammalian sleep (Krishnan et al., 2019; Louie & Wilson, 2001). Recent evidence in biology (McClelland et al., 2020) shows that only a few selected experiences are replayed during sleep, in contrast

[1]SRI International, Princeton, NJ, USA. Correspondence to: Aswin Raghavan <aswin.raghavan@sri.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

to the typical replay mechanisms in machine learning. Our proposed approach bridges these gaps.

Our main contribution is a framework that combines generative memory with a set of skills that span the space of behaviors necessary to solve any given task. The framework consists of a set of generator-skill pairs that partition a stream of data into what we call *eigentasks*. Each generator models a subset of the input space, and the corresponding skill encodes the appropriate outputs for the inputs in the generator's support set. The likelihood of an input according to each generator is used to retrieve the appropriate skill, making the eigentask model a content-addressable memory for skills and enabling forward transfer. Eigentasks can be seen as a combination of generative memory with mixture-of-experts (MoE) models (Makkuva et al., 2018; Tsuda et al., 2020). The MoE component facilitates forward transfer, while the generative component avoids forgetting.

We develop a concrete instantiation of eigentasks called Open World Variational Auto Encoder (OWVAE) that uses a set of VAEs (Kingma & Welling, 2014) as generators. OWVAE partitions data into eigentasks using out-of-distribution detection based on a likelihood ratio test in the latent space of the generator. We present a loss function for end-to-end learning of eigentasks that incorporates the losses of the generators and skills, weighted by the likelihood ratio. We show experimentally that OWVAE achieves superior performance comapred to state-of-the-art (SOTA) generative memory (GM) approaches on a new benchmark that contains a mix of MNIST and FashionMNIST datasets, and comparable performance in the splitMNIST benchmark. OWVAE's superior performance is attributed to task disentanglement (Achille et al., 2018) and confirmed visually by comparing the samples generated by each VAE.

Our second contribution is a sampling strategy that improves the quality of generative replay by using the confidence of the predictions output by the paired skills to reject out-of-distribution samples. Our experiments show improved continual learning and reduction in forgetting when using rejection sampling as compared to accepting all generated examples.

Our third contribution is a lifelong RL algorithm that leverages the OWVAE for exploration of new tasks. The OWVAE generates advice by recalling options (Sutton et al., 1999) that are applicable to the current task, based on the eigentask partitioning of old tasks. This advice is incorporated into an off-policy learner whose behavior policy is a mixture of the options and the policy being learned. We apply our lifelong RL algorithm to the challenging video game Starcraft 2 (SC2). On a sequence of "mini-games" (Vinyals et al., 2017) as tasks, our approach shows positive forward transfer when the new task is of the same type as one of the old tasks. In most mini-games we observe a jump start in the accumulated reward, and in one mini-game, forward transfer outperforms the single-task learner by 1.5x with 10x fewer samples due to better exploration.

2. Background

In lifelong learning, an agent is faced with a never-ending sequence of *tasks*. Each task i is defined by the tuple $T_i = (\mathcal{X}_i, \mathcal{Y}_i, P_i, L_i)$, where $\mathcal{X}_i$ is the input space, $\mathcal{Y}_i$ is the output space, $P_i \in \text{Prob}(\mathcal{X}_i)$ is the input distribution, and $L_i(\mathbf{x}, \mathbf{y}) : \mathcal{X}_i^* \times \mathcal{Y}_i^* \mapsto \mathbb{R}$ gives the loss for outputting $\mathbf{y} = \{y_1, \ldots, y_k\}$ for inputs $\mathbf{x} = \{x_1, \ldots, x_k\}$. In this paper we consider the case where all the input and output spaces are the same: $\mathcal{X}_i \equiv \mathcal{X}$, $\mathcal{Y}_i \equiv \mathcal{Y}$, and our definition of a task is $T_i = (P_i, L_i)$. We denote the (countably infinite) set of tasks by $\mathcal{T}$. Since a task T_i is totally characterized by its index i, we use tasks and their indices interchangeably.

A task sequence is a sequence of tuples $((i_1, n_1), (i_2, n_2), \ldots)$ drawn from the task distribution W, where each $i_k \in \mathcal{T}$ is a task index and the corresponding n_k is the number of samples from P_{i_k} that the agent sees before the transition to the next task. The lifelong learning agent's objective is to learn a single hypothesis $h : \mathcal{X} \mapsto \mathcal{Y}$ that minimizes the expected per-sample loss wrt the task distribution,

$$\mathcal{L}[h] = E_{(i,n)\sim W} E_{\mathbf{X}^n \sim P_i}[L_i(\mathbf{X}^n, h(\mathbf{X}^n))]. \quad (1)$$

where $\mathbf{X}^n$ are n instances sampled IID from P. Equation 1 defines the optimal hypothesis wrt the task distribution. Note that the task distribution W is unknown to the agent, and $\mathbf{X}^n$ does not contain the task index. Lifelong learning (Eq. 1) is strictly harder than multi-task learning, where instances have known task IDs.

2.1. Backward and Forward Transfer

The notion of knowledge transfer is central to evaluating success in lifelong learning. In this paper, we approach lifelong learning in an episodic setting where the agent learns in a series of learning epochs of fixed length (not necessarily aligned with task boundaries). In this setting, we can define the key metrics of forward and backward transfer in a manner that is agnostic to the task boundaries. Let N_i^j be the total number of samples seen from task i through the end of epoch j. We define a loss restricted to tasks actually seen by the learner through epoch j as

$$\ell^j[h] = \sum_{i \in \mathcal{T}} \mathbb{1}(N_i^j > 0) E_{X \sim P_i}[L_i(X, h(X))] \quad (2)$$

Backward Transfer (BT) describes the difference in performance on old tasks before and after one or more epochs of learning. Let $h^{j:k}$ be the hypothesis obtained after training

sequentially on the data from epochs $j, j+1, \ldots, k$, where h^j is shorthand for $h^{j:j}$. The one-step backward transfer is,

$$\text{BT}(j) = \ell^{j-1}[h^{1:j}] - \ell^{j-1}[h^{1:j-1}]. \quad (3)$$

Negative BT corresponds to forgetting knowledge learned in previous episodes, and positive BT could indicate successful knowledge consolidation between tasks. Forward Transfer (FT) describes the difference in loss on new tasks with and without training on previous tasks.

$$\text{FT}(j) = (\ell^j[h^{1:j}] - \ell^{j-1}[h^{1:j}]) - (\ell^j[h^j] - \ell^{j-1}[h^j]). \quad (4)$$

The terms within parentheses restrict the loss to new tasks in the jth episode. Positive FT indicates transfer of knowledge or skills between tasks, and typically corresponds to jump start performance and lower sample complexity.

2.2. Reinforcement Learning

The flexibility of our eigentask framework allows it to be applied in supervised learning (SL), unsupervised learning, and reinforcement learning (RL). For simplicity, we describe lifelong RL in the finite Markov decision process (MDP) setting. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \mapsto \text{Prob}(\mathcal{S})$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function. The objective is a policy $\pi : \mathcal{S} \mapsto \text{Prob}(\mathcal{A})$ that maximizes the value function $V^\pi(s) = E_{S_t \sim P, A_t \sim \pi}[\sum_{t=1}^{\infty} \gamma^t R(S_t, A_t)]$. Tasks in lifelong RL are MDPs, with common state and action spaces, but P and R may differ. The loss is the negative return, $L(\mathbf{x}, \mathbf{y}) = -\sum_{(x,y)\in(\mathbf{x},\mathbf{y})} R(x, y)$.

3. Eigentask Framework

Lifelong learning agents must balance plasticity vs. stability: improving at the current task while maintaining performance on previously-learned tasks. The problem of stability is especially acute in neural network models, where naïvely training an NN model on a sequence of tasks leads to *catastrophic forgetting* of previous tasks. A proven technique for avoiding forgetting is to mix data from all previous tasks with data from the current task during training, thereby reverting the streaming learning problem back to an offline learning problem. The generative memory (GM) approach accomplishes this with a generative model of the data distribution $p(x, y)$ factorized as a generator $p(x)$ over the input space $\mathcal{X}$ and a discriminator or skill $p(y|x)$ conditioned on the input x. A skill can correspond to a classifier in supervised learning (SL) or to a policy or option in RL.

To achieve selective transfer, we propose to use multiple generator-skill pairs to disentangle streaming data into "canonical tasks" that we call *eigentasks*. Informally, eigentasks partition the joint input-output space such that all

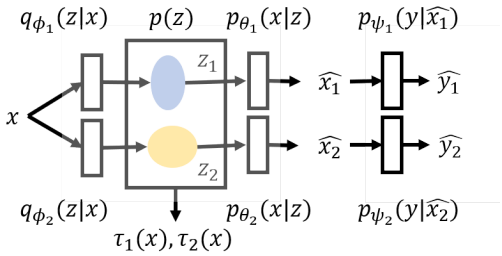


Figure 2. An Open World Auto Encoder (OWVAE) with two eigentasks, showing the terms used in the loss function (Eq. 7).

inputs within an eigentask use the same skill. Eigentasks capture task similarity defined in terms of the combination of generative and skill losses. The use of multiple generators corresponds to a mixture model. Prior work on mixtures of VAEs or GANs (Dilokthanakul et al., 2016; Zhang et al., 2017; Rao et al., 2019) cluster inputs based on perceptual similarity alone or create one task per label. On the other hand, mixtures-of-experts (Tsuda et al., 2020) capture skill similarity alone. The eigentask framework combines the MoE concept with generative replay, to avoid forgetting and realize selective transfer.

3.1. Eigentask Definition

Formally, an eigentask is a tuple $E = (g, f)$ comprising a generator $g(\epsilon)$ that defines a distribution over inputs $\mathcal{X}$ as a function of random noise ϵ, and a skill $f : \mathcal{X} \mapsto \text{Prob}(\mathcal{Y})$ that maps inputs to outputs. An eigentask model $M = (\mathcal{E}_n, \tau)$ consists of a set of eigentasks $\mathcal{E}$ and a similarity function $\tau : \mathcal{X} \mapsto \text{Prob}(\mathcal{E})$ whose output is a probability vector over the n eigentasks. Typically, g, f and τ are parameterized functions such as DNNs. We write τ as a function of the current input, but in general it may be a function of the entire input history. We first describe the loss function for end-to-end learning of eigentasks in the offline setting. The extension to the streaming section is discussed in the next section. Given a dataset $\{X, Y\}$, the general loss function for eigentask learning is,

$$\min \sum_{i=1}^{n} \tau_i(X)[\mathcal{L}_{gen}(g_i(\epsilon)|X) + \mathcal{L}_{disc}(f_i|g_i(\epsilon), X, Y)] \quad (5)$$

where $\mathcal{L}_{gen}$ and $\mathcal{L}_{disc}$ denote generative and discriminative losses for the generator and skill, respectively. The loss in (5) is agnostic to the choice of generator (e.g., VAE or GAN), skill (e.g. classifier, policy, options), and similarity function τ. Note that the generator-skill pairs are independent, to avoid interference between eigentasks. This is in contrast to recent approaches that learn a shared embedding across tasks (e.g. Achille et al., 2018). In our experiments, $\mathcal{L}_{disc}$ is a cross-entropy loss for classification problems and an RL loss such as the policy gradient for RL problems.

We develop a concrete instantiation of eigentasks called

Open World Variational Auto Encoder (OWVAE) that uses a set of VAEs (Kingma & Welling, 2014) as generators, with latent space denoted by z, encoder $q_\phi(z|x)$, decoder $p_\theta(x|z)$ and prior $z \sim N(0, \mathbf{I})$. Figure 2 shows an OWVAE with two eigentasks. We use reconstruction error between the input and decoder output as the generative loss $\mathcal{L}_{gen}$. We propose to use a likelihood ratio test to define $\tau(x)$. [1] We use the likelihood of decoder i generating the observed data x for some z, that is approximated using encoding z_i.

$$\tau_i(x) = \sigma\Big(\frac{p_{\theta_i}(x|z)}{\max_j p_{\theta_j}(x|z)}\Big) \approx \sigma\Big(\frac{\Phi(z_i)}{\max_j \Phi_j(z_j)}\Big) \quad (6)$$

where subscript i denotes eigentask i, $z_i = q_{\phi_i}(z|x)$ is the encoding, Φ is the density of standard gaussian, σ denotes softmax. The assumption is valid when the decoder weights are the inverse of the encoder weights. The loss function for OWVAE is in Eq. 7,

$$\min_{\theta,\phi,\psi} E_{x,y}[E_{\tau(x)}[\mathcal{L}_{VAE}(x;\theta,\phi) + \log p_\psi(y|\hat{x})]] \quad (7)$$

where $\mathcal{L}_{VAE}$ is the standard VAE loss,

$$\mathcal{L}_{VAE}(x;\theta,\phi) = E_{q_\phi(z|x)} \log p_\theta(x|z) - D_{KL}(q_\phi(z|x)||p(z)).$$

At test time, given an input x we calculate $\tau(x)$ by computing a forward pass through the encoder. We sample the index of the decoder and skill to use according to the categorical distribution given by $\tau(x)$. In the simplest case, we pass the decoder output as the input to the skill. In our experiments, we observed that using the mid-level features (the encoder features before projecting to latent space) led to the best accuracy of the skill, matching an observation by van de Ven & Tolias (2018). Note that the above sampling method is conditional on x, and that the OWVAE does not allow direct sampling from the learned mixture because the mixing coefficient $\tau(x)$ depends on the input. Generative replay can be used to learn an OWVAE incrementally but requires direct sampling of old tasks. Section 4 proposes a strategy for direct sampling.

3.2. The Wake-Sleep Cycle

Our lifelong learning agents operate in a wake-sleep cycle, solving a stream of tasks during the wake phase, and consolidating new task knowledge during the sleep phase using generative replay. In the wake phase, the learner's goal is to maximize FT wrt correctly outputs on incoming inputs. In supervised learning that might mean simply outputting the correct label according to the new task, while in reinforcement learning the agent needs to explore the new task and maximize reward. During the wake phase, the learner converts the streaming input to batched data, by storing new task examples in a short-term buffer along with any intermediate solutions of wake phase learning. Periodically, the learner enters a sleep phase, where the objective is memory consolidation over all tasks with minimal negative BT. We use generative replay to incorporate new task batches in to an eigentask model that is continuously updated. A general wake-sleep cycle is shown in Algorithm 1 where sleep phase is activated whenever the buffer is full. We describe instantiations of the wake-sleep cycle for supervised learning and RL in the next two sections.

[1] (Ren et al., 2019) developed a similar LR-test concurrently.

Algorithm 1 General Wake-Sleep Cycle

Input: OWVAE M, buffer B, # sleep iterations N
Initialize M; set B to empty.
while True **do**
  Initialize task learner T
  repeat {Wake Phase}
    Classification (Section 4): store new instance in buffer
    RL (Section 5): Update T, B with Alg 3
  until B is full
  Create copy $M_2 = M$
  for N iterations **do** {Sleep Phase}
    Fetch batch b from B
    Generate replay $b_M \sim M_2$ using Alg 2
    Update M using Eq. 7 on $b \cup b_M$
  end for
  Set B to empty
end while

4. Lifelong Supervised Learning

Wake phase: In this work, we use a trivial wake phase for supervised classification. We simply store the new task examples (instance and label) to the buffer. The OWVAE could be leveraged in the wake phase, e.g. augment new task data with selective replay of similar tasks, or using the OWVAE skills as hints for knowledge distillation. For example, hints led to positive FT when the new task was a noisy version of old tasks (noise added to labels and pixels). We have not yet investigated these possibilities completely as they are specific to the scenario. Algorithms for learning from streaming data (e.g. Hayes et al., 2018; Smith et al., 2019) can be used to update the task learner.

Sleep phase: As mentioned in Section 3.1, sampling from an OWVAE is conditional on input x because the mixing coefficients $\tau(x)$ are a function of x. Some eigentasks may have received little or no training (e.g. when old tasks are few and similar). These untrained generators will generate noise that must not be used to augment new task data. To mitigate this problem, we developed a rejection sampling strategy using the confidence of the skill associated with

Algorithm 2 Rejection Sampling from OWVAE

Input: OWVAE M, batch size K, threshold δ
Initialize deque $D[y]$ for each label y of size K
repeat
  for Eigentask i in M **do**
    Generate $z_i \sim N(0, \mathbf{I})$, $x_i \sim p_{\theta_i}(x|z_i)$
    $y_i = \arg\max p_{\psi_i}(y|x_i)$, $\text{conf}_i = \max p_{\psi_i}(y|x_i)$
    if $\text{conf}_i > \delta$: Push x_i, y_i to $D[y_i]$
  end for
until Each $D[y]$ is full or MAX_TRIES
Return $\bigcup_y D[y]$

Algorithm 3 Exploration using OWVAE for lifelong RL

Input: OWVAE M, Buffer B, Off-policy learner A, MDP $\mathbb{M}$
Initialize policy π, $\mu(t)$
repeat
  $s =$ state from $\mathbb{M}$, get $\tau(s)$ from OWVAE using Eq. 6
  Sample action a using $b(s)$ and $\tau(s)$ as in Eq. 8
  Execute a in $\mathbb{M}$. Observe next state s' and reward r
  Add (s, a, s', r) to RL training set
  Update π using A. Decrease $\mu(t)$
until Sample budget reached
Reset $\mathbb{M}$ to initial state
Execute π to generate set of (s, a, s', r)
Add (s, a, s', r) to buffer B

the generator. Algorithm 2 shows the sampling strategy for OWVAE. We reject a sample (x, y) if the confidence of the associated skill is below a threshold δ. We reject over-represented labels and generate label-balanced replay. These two refinements to the sampling process were critical in achieving high accuracy on continual learning benchmarks. In addition, rejection sampling improved the accuracy of the basic GM approach as well.

5. Lifelong Reinforcement Learning

In the lifelong RL setting, tasks are MDPs (Section 2.2), the eigentask skills are policies, and the associated generators generate states where the policies should be applied.

Wake Phase: One of the key determinants of RL performance is the efficiency of exploration. Without any prior knowledge, RL algorithms typically explore randomly in the early stages of learning. Our approach (see Algorithm 3) is to use the skill corresponding to the most relevant eigentask to aid in exploration. An off-policy RL algorithm is used for training because the exploration policy is defined by these skills. The behavior policy is a mixture of the target policy and the mixture of skills induced by the OWVAE's τ-function. Let π denote the target policy and let p_{ψ_i} be the ith skill. Given a state s, the behavior policy $b(s)$ is,

$$b(s) = \begin{cases} \pi(s) & \text{w.p. } 1 - \mu(t) \\ p_{\psi_i}(s) & \text{w.p. } \mu(t)\tau(s)_i \text{ each eigentask } i \end{cases} \quad (8)$$

where $\mu(t) \in [0, 1]$ is a "mixing" function that decays over time so that eigentask usage is gradually reduced and replaced by π. We use importance weighting as implemented by the off-policy actor-critic algorithm VTrace (Espeholt et al., 2018), but our approach is compatible with any off-policy algorithm. In the last step of the wake phase, trajectories from the target policy are stored in the buffer, to be later consolidated in the sleep phase.

Sleep Phase: In the sleep phase, the goal is to consolidate the final target policy π into the eigentask skills p_{ψ_i}. Our approach is based on policy distillation (Rusu et al., 2015), that transforms the problem to a supervised learning problem. The sleep phase proceeds in the same manner as for supervised learning tasks (Section 4).

6. Experiments

We validate the idea of eigentasks on unsupervised, supervised classification and RL tasks. We show experimentally that OWVAE achieves superior performance comapred to state-of-the-art (SOTA) generative memory (GM) approaches RtF (van de Ven & Tolias, 2019) and DGR (Shin et al., 2017) on a new benchmark that contains a mix of MNIST and FashionMNIST datasets, and comparable performance in the splitMNIST benchmark for continual learning. OWVAE's superior performance is attributed to task disentanglement (Achille et al., 2018) and confirmed visually by comparing the samples generated by each VAE. We demonstrate our lifelong RL algorithm on the Starcraft 2 (SC2) mini-games benchmark (Vinyals et al., 2017). We demonstrate that our approach compares favorably to the baselines of single-task and multi-task RL.

6.1. Illustration: synthetic problem

In this section, we illustrate the OWVAE model on a synthetic but challenging problem for current GM approaches. The problem is inspired by the Wisconsin Card Sorting task (Tsuda et al., 2020), where the tasks cannot be distinguished by "perceptual similarity", but can be distinguished by "skill similarity", so a mixture-of-experts model can solve the tasks (Tsuda et al., 2020). We test whether an OWVAE can separate the tasks and achieve a high accuracy.

Consider two binary classification tasks whose input space is an isotropic Gaussian in two dimenstions, and the labels for the two tasks are flipped, e.g. task-0 labels are given by $(X \geq 0)$, and task-1 labels are $(X < 0)$. The data distribution is shown in Figure 3a. Any classifier that achieves an accuracy of $p \in [0, 1]$ on task-0 must have accuracy $1 - p$ on

task-1, and thus average accuracy of 0.5. Figure 3b shows that an OWVAE with two eigentasks is able to achieve high accuracy (> 0.7). Figure 3c shows the reason: a meaningful τ has been learned that separates the tasks successfully into two eigentasks. Interestingly, the learned eigentasks are $(X \geq 0)$ and $(X < 0)$, i.e. label-0 of task-0 is combined with label-1 of task-1 within one eigentask, and label-1 of task-0 is combined with label-0 of task-1 within another eigentask. In contrast, current GM methods will not work because learning either task causes forgetting of the other.

6.2. Continual Learning for Supervised Classification

We use the class-incremental learning (Class-IL) setting introduced in (van de Ven & Tolias, 2019). In this setting, new classes or groups of classes are presented incrementally to the learning alorithm. We use the standard splitMNIST problem and compare to the SOTA. In splitMNIST, the MNIST dataset is split into five tasks with each task having two of the original classes. Further, we create a new benchmark combining MNIST and Fashion-MNIST datasets. The combined MNIST and Fashion-MNIST classes are split evenly into ten tasks. Each task introduces two new classes, one MNIST digit and one fashion article. In this new benchmark, we establish a new SOTA by showing that current replay based approaches are inferior to OWVAE.

In both benchmarks, each new task is trained for 500 iterations with a batch size of 32. All compared SOTA approaches use a two layer perceptron with 650 neurons for the encoder and decoder, and a latent dimension of 100. We compare OWVAE against eight continual learning approaches spanning regularization, replay, and replay with exemplars (Table 1). In addition, we show two baselines of single-task learner (lower bound - only learns on current task) and offline multi-task learner (upper bound - knows all tasks). Training is done on the standard train sets and results are reported on the standard test sets.

The OWVAE uses two eigentasks, each with the same architecture as SOTA but with 400 neurons only, so that the OWVAE has the same number of parameters as SOTA. Within the OWVAE, the inputs to the skill are the mid-level features of the corresponding encoder, i.e. activations in the last layer of the encoder. As in Section 4, no wake phase is used and sleep phase uses 500 iterations with a batch size of 32, and the threshold δ is set to 0.5 (as in Alg. 2). To study the impact of rejection sampling, we perform an ablation study over different augmentation strategies: (1) BaseAug: all examples generated during replay are accepted, (2) BAug: rejection sampling to create label-balanced replay, (3) VAug: rejecting low confidence examples, and (4) VBAug: combining (2) and (3) (as in Alg 2).

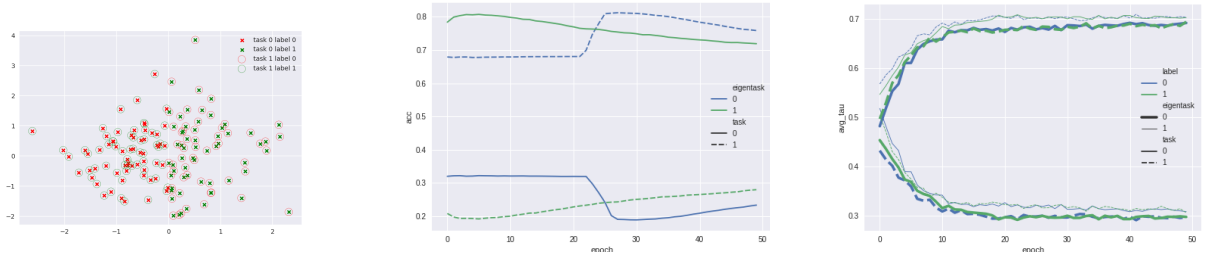
The average accuracies over all tasks at the end of training are shown in Table 1. As observed in prior work, the class-IL setting is hard for the regularization approaches like EWC, as well as LwF; their performance is very low, comparable to the single task learner (they learn and immediately forget each task). Our approach (Replay+Eigentasks) has accuracy comparable to other replay-based methods (except LwF) on splitMNIST. However, on split(MNIST+FashionMNIST), our approach has a higher accuracy (about 4% higher). Unsurprisingly, using exemplars within replay improves accuracy on both benchmarks. Exemplars could be integrated into OWVAE in future work.

Approach	Method	D1	D2
Baselines	None - lower bound	19.90	10.22
	Offline - upper bound	97.94	90.89
Regularization	EWC	20.01	10.00
	Online EWC	19.96	10.00
	SI	19.99	10.00
Replay	LwF	23.85	10.07
	DGR	90.79	73.36
	DGR x2	91.83	65.82
	DGR+distill	91.79	72.40
	DGR+distill x2	94.01	67.37
Replay+Exemplars	RtF	92.56	61.15
	RtF x2	92.86	61.41
	iCaRL	94.57	82.69
Replay+Eigentask	ET1-BaseAug	87.68	69.29
	ET1-BAug	90.99	74.11
	ET1-VAug	87.33	63.34
	ET1-VBAug	90.69	77.43
	ET2-BaseAug	88.93	57.91
	ET2-BAug	**91.27**	69.95
	ET2-VAug	82.08	69.55
	ET2-VBAug	90.25	**76.81**

Table 1. Average test accuracy over all tasks on splitMNIST (D1) and split(MNIST+FashionMNIST) (D2) benchmarks. ET1 and ET2 denote the number of eigentasks in an OWVAE model. Methods compared: EWC (Kirkpatrick et al., 2016), Online EWC (Schwarz et al., 2018), SI (Zenke et al., 2017), LwF (Li & Hoiem, 2016), DGR(Shin et al., 2017), RtF (van de Ven & Tolias, 2019),and iCaRL (Rebuffi et al., 2017). The variants denoted x2 have the same number of parameters as ET2.

The ablation study shows that VBAug (as in the combination of rejection using both confidence and label) leads to the most improvement (2-3% on SplitMNIST, 19% on split(MNIST+FashionMNIST)). The combination VBAug performs better than BAug, BaseAug and VAug, whereas VAug by itself seems to decrease the performance wrt BaseAug. Interestingly, VBAug and BAug improved the performance of the basic GM approach (as in BaseAug vs ET1: 2-3% improvement on splitMNIST, 5-7% improvement on split(MNIST+FashionMNIST)). A detailed breakdown of accuracy per-task over time is shown in the appendix.

Figure 4 shows the task separation learned by OWVAE. The figure visualizes VAE reconstructions for each task. It shows that the first eigentask has learned all the MNIST digits and able to reconstruct them, whereas the second eigentask has



(a) Setup of the synthetic problem with conflicting tasks.

(b) Accuracy with OWVAE(2): both tasks can be separated and learned.

(c) Tracking τ shows correctly identified eigentasks.

Figure 3. Illustration on synthetic problem (Section 6.1): OWVAE(2) learns two conflicting tasks with no perceptual dissimilarity.

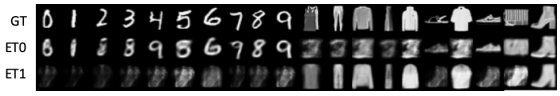


Figure 4. Split(MNIST+FashionMNIST): Image reconstruction by OWVAE(2) showing task separation. Top row: Ground truth. Middle and Bottom: reconstruction by first and second eigentask.

learned all the fashion articles. The first eigentask has also learned some fashion articles whereas the second eigentask has not learned any digits. The blurry and noisy images are removed from replay by our rejection sampling strategy.

6.3. Starcraft 2

We use the Starcraft 2 learning environment (SC2) (Vinyals et al., 2017). SC2 is a rich platform in which diverse RL tasks can be implemented. We used their "mini-games" as the task set for our experiments. Our policy architecture is a slight modification of their FullyConv architecture. We use the 17 64x64 feature maps extracted by SC2LE.

6.3.1. Eigentask Learning

In the OWVAE, we use only two feature maps namely "unit type" (identity of the game unit present in each pixel) and "unit density" (average number of units per pixel). An example of these feature maps is shown in the appendix.

The eigentasks for SC2 are learned in an unsupervised and offline manner. We first collected a dataset by executing a random policy in each mini-game and recording the feature maps per frame. Whenever possible, we also collected similar data by running a scripted agent. We then trained an OWVAE incrementally with three eigentasks in a fully unsupervised manner. The setup is similar to the continual learning experiment for splitMNIST (each mini-game is seen for 500 iterations etc.). SC2 mini-games are perceptually different, so the OWVAE is able to separate the tasks into meaningful eigentasks despite the unsupervised training. As shown in Figure 5, by looking at the relative values of the OWVAE τ-function, we see that the first eigentask grouped all combat tasks together, but incorrectly grouped a

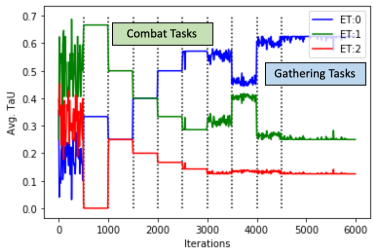
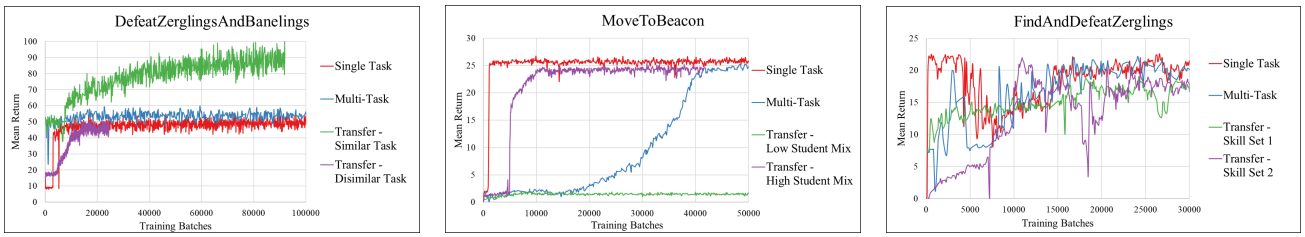


Figure 5. Continual Unsupervised Learning of a OWVAE(3) in SC2 mini-games: Variation of τ over iterations and learned task grouping. Each mini-game is observed for 500 iterations.

navigation task (possibly due to the unsupervised training). The second eigentask learned the BuildMarines task alone, whereas the third eigentask grouped the resource gathering tasks together. For each task type, we observe one eigentask clearly dominating the τ values, while no single eigentask dominates always. These task groupings can be confirmed by looking at the VAE reconstructions shown in appendix.

6.3.2. Forward Policy Transfer

We focus on the wake phase of our lifelong RL algorithm (Alg 3), and examine forward transfer due to efficient exploration. We manually set the OWVAE skills selected from the set of trained single-task policies. These policies are separated into groups, based on the task separation observed from the unsupervised OWVAE training above. Each skill is assigned a policy from the corresponding group. Training uses the VTrace learning rule (Espeholt et al., 2018) in an A2C implementation heavily adapted from code published by Ring (2018). We examined transfer from skills from source tasks that are either similar or dissimilar to the target task (Table 2). The main results with forward transfer are summarized in Figure 6. The plots include two baselines: the performance of a single task policy trained on the target task, and a multi-task policy trained on batches containing experience from all 6 tasks. In order to demonstrate efficient exploration, the vertical axis shows mean per-episode return obtained by the behavior policy during training.



(a) (b) (c)

Figure 6. Forward transfer in Starcraft 2 mini-games.

Category	**Tasks**
Combat	DefeatRoaches, DefeatZerglingsAndBanelings
Navigation	MoveToBeacon, CollectMineralShards
Hybrid	FindAndDefeatRoaches
Economy	CollectMineralsAndGas

Table 2. Categories of SC2 tasks. Tasks in the same category are considered "similar" in our experiments.

Experiment (6a) examines transfer to the DefeatZerglingsAndBanelings task. In the Transfer-Similar Task condition, the OWVAE skills are policies trained on CollectMineralShards and DefeatRoaches. The DefeatRoaches task is another combat task, and thus is similar to the target task. In the Transfer-Disimilar Task condition, the OWVAE skills are CollectMineralShards and MoveToBeacon. In the Transfer-Similar condition, our approach yielded both good performance from the start of training and substantially better asymptotic performance. Interestingly, our approach even surpassed the asymptotic performance of single-task and multi-task learning, clearly showing the impact of better exploration transferred through the OWVAE skills. Furthermore, the asymptotic performance is also better than the best published performance for the FullyConv policy architecture (Vinyals et al., 2017) by about 1.5x while using 10x fewer RL iterations. In the Transfer-Dissimilar condition, our approach still resulted in better initial performance than single task training, but converged to the asymptotic performance of the single task policy at a slower rate.

In experiment (6b), we study transfer to the MoveToBeacon task. Because of the way we trained the OWVAE, the MoveToBeacon task gets clustered with two Combat tasks, and not with the more similar CollectMineralShards task. As a result, the OWVAE τ function selects a combat skill to use for transfer to MoveToBeacon. When we use the default schedule for the behavior policy mixing rate $\mu(t)$, transfer from the inappropriate skill hinders learning. However, using a different mixing schedule that gives more weight to the target policy allows the agent to overcome the effect.

Finally, experiment (6c) investigates transfer to the FindAndDefeatZerglings task. This is an interesting target task because it combines elements of Combat and Navigation tasks, but is not highly similar to either of those categories. We evaluated two different skill sets (either CollectMineralShards+DefeatRoaches or CollectMineralShards+MoveToBeacon) but transfer had no clear positive or negative effect for this target task.

7. Discussion and Future Work

We introduced the eigentask framework for lifelong learning, which combines generative replay with mixture-of-experts style skill learning. We use the framework in a wake-sleep cycle where new tasks are solved in the wake phase and experiences are consolidated into memory in the sleep phase. We applied it to both lifelong supervised learning and RL problems. We developed refinements to the standard generative replay approach to enable selective knowledge transfer. Combined with the rejection sampling trick, we achieved SOTA performance on continual learning benchmarks. In lifelong RL, we demonstrated successful forward transfer to new tasks in Starcraft 2, and exceeded the best published performance on one of the Starcraft 2 tasks.

Our immediate goal in future work is to close the wake-sleep loop in lifelong RL. We have demonstrated success for components of the approach, but not for the full eigentask framework. We are interested in adding change-point detection to improve on the likelihood ratio test, and hierarchical eigentasks that could be more compact and more efficient. Finally, we want to incorporate task similarity measures that account for history, to separate RL tasks that have similar observations but different dynamics.

Acknowledgements

This material is based on work supported by the Lifelong Learning Machines (L2M) program of the Defense Advanced Research Projects Agency (DARPA) under contract HR0011-18-C-0051. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the Department of Defense or the U.S. Government.

References

Achille, A., Eccles, T., Matthey, L., Burgess, C., Watters, N., Lerchner, A., and Higgins, I. Life-long disentangled representation learning with cross-domain latent homologies. In *Advances in Neural Information Processing Systems*, pp. 9873–9883, 2018.

Ammar, H. B., Eaton, E., Luna, J. M., and Ruvolo, P. Autonomous Cross-Domain Knowledge Transfer in Lifelong Policy Gradient Reinforcement Learning. In *IJCAI*, 2015.

Chen, Z. and Liu, B. Lifelong Machine Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(3), 2016. ISSN 1939-4608, 1939-4616. URL www.morganclaypool.com/doi/10.2200/S00737ED1V01Y201610AIM033.

Diekelmann, S. and Born, J. The memory function of sleep. *Nature Reviews Neuroscience*, 11(2), 2010.

Dilokthanakul, N., Mediano, P. A. M., Garnelo, M., Lee, M. C. H., Salimbeni, H., Arulkumaran, K., and Shanahan, M. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *arXiv:1611.02648 [cs, stat]*, 2016. URL arxiv.org/abs/1611.02648.

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

Fei, G., Wang, S., and Liu, B. Learning Cumulatively to Become More Knowledgeable. In *ACM SIGKDD '16*, San Francisco, California, USA, 2016. ACM Press. ISBN 978-1-4503-4232-2. URL dl.acm.org/citation.cfm?doid=2939672.2939835.

Hayes, T. L., Cahill, N. D., and Kanan, C. Memory Efficient Experience Replay for Streaming Learning. *arXiv preprint arXiv:1809.05922*, 2018.

He, C. Exemplar-Supported Generative Reproduction for Class Incremental Learning. In *British Machine Vision Conference (BMVC)*, 2018.

Huszár, F. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 2018.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, 2014. URL arxiv.org/abs/1312.6114.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., 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. Overcoming catastrophic forgetting in neural networks. *arXiv:1612.00796 [cs, stat]*, 2016. URL arxiv.org/abs/1612.00796.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., and others. Reply to Huszár: The elastic weight consolidation penalty is empirically valid. *Proceedings of the National Academy of Sciences*, 115(11), 2018.

Krishnan, G. P., Tadros, T., Ramyaa, R., and Bazhenov, M. Biologically inspired sleep algorithm for artificial neural networks. *arXiv:1908.02240 [cs]*, 2019. URL arxiv.org/abs/1908.02240.

Lee, S., Stokes, J., and Eaton, E. Learning Shared Knowledge for Deep Lifelong Learning using Deconvolutional Networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Macao, China, 2019. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-4-1. URL https://www.ijcai.org/proceedings/2019/393.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting. *arXiv:1904.00310 [cs]*, 2019. URL arxiv.org/abs/1904.00310.

Li, Z. and Hoiem, D. Learning without Forgetting. *arXiv:1606.09282 [cs, stat]*, 2016. URL arxiv.org/abs/1606.09282.

Liu, X., Yang, H., Ravichandran, A., Bhotika, R., and Soatto, S. Continual Universal Object Detection. *arXiv:2002.05347 [cs]*, 2020. URL arxiv.org/abs/2002.05347.

Louie, K. and Wilson, M. A. Temporally structured replay of awake hippocampal ensemble activity during rapid eye movement sleep. *Neuron*, 29(1), 2001.

Makkuva, A. V., Oh, S., Kannan, S., and Viswanath, P. Breaking the gridlock in Mixture-of-Experts: Consistent and Efficient Algorithms. *arXiv:1802.07417 [cs]*, 2018. URL arxiv.org/abs/1802.07417.

Masse, N. Y., Grant, G. D., and Freedman, D. J. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proc Natl Acad Sci USA*, 115 (44), 2018. ISSN 0027-8424, 1091-6490. URL arxiv.org/abs/1802.01569.

McClelland, J. L., McNaughton, B. L., and Lampinen, A. K. Integration of New Information in Memory: New Insights from a Complementary Learning Systems Perspective. preprint, Neuroscience, 2020. URL biorxiv.org/lookup/doi/10.1101/2020.01.17.909804.

Nagabandi, A., Finn, C., and Levine, S. DEEP ONLINE LEARNING VIA META-LEARNING: CONTINUAL ADAPTATION FOR MODEL-BASED RL. In *ICLR*, 2019.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational Continual Learning. *arXiv:1710.10628 [cs, stat]*, 2018. URL arxiv.org/abs/1710.10628.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual Lifelong Learning with Neural Networks: A Review. *Neural Networks*, 113, 2019. ISSN 08936080. URL arxiv.org/abs/1802.07569.

Ramapuram, J., Gregorova, M., and Kalousis, A. Lifelong Generative Modeling. *arXiv:1705.09847 [cs, stat]*, 2017. URL arxiv.org/abs/1705.09847.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems*, 2019.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., DePristo, M. A., Dillon, J. V., and Lakshminarayanan, B. Likelihood Ratios for Out-of-Distribution Detection. *arXiv:1906.02845 [cs, stat]*, 2019. URL arxiv.org/abs/1906.02845.

Ring, R. Reaver: Modular deep reinforcement learning framework. https://github.com/inoryy/reaver, 2018.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive Neural Networks. *arXiv:1606.04671 [cs]*, 2016. URL arxiv.org/abs/1606.04671.

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.

Silver, D. L., Yang, Q., and Li, L. Lifelong Machine Learning Systems: Beyond Learning Algorithms. In *AAAI Spring Symposium Series*, 2013.

Smith, J., Baer, S., Kira, Z., and Dovrolis, C. Unsupervised continual learning and self-taught associative memory hierarchies. 2019.

Sutton, R. S., Precup, D., and Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2): 181–211, 1999.

Tessler, C., Givony, S., Zahavy, T., Mankowitz, D. J., and Mannor, S. A Deep Hierarchical Approach to Lifelong Learning in Minecraft. *arXiv:1604.07255 [cs]*, 2016. URL arxiv.org/abs/1604.07255.

Tsuda, B., Tye, K. M., Siegelmann, H. T., and Sejnowski, T. J. A modeling framework for adaptive lifelong learning with transfer and savings through gating in the prefrontal cortex. *bioRxiv*, 2020. URL https://www.biorxiv.org/content/early/2020/03/12/2020.03.11.984757.

van de Ven, G. M. and Tolias, A. S. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

van de Ven, G. M. and Tolias, A. S. Three scenarios for continual learning. *arXiv:1904.07734 [cs, stat]*, 2019. URL arxiv.org/abs/1904.07734.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al. Starcraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Wilson, M. A. and McNaughton, B. L. Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172), 1994.

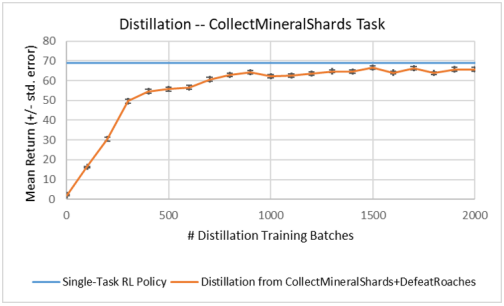
Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, 2014.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3987–3995. JMLR. org, 2017.

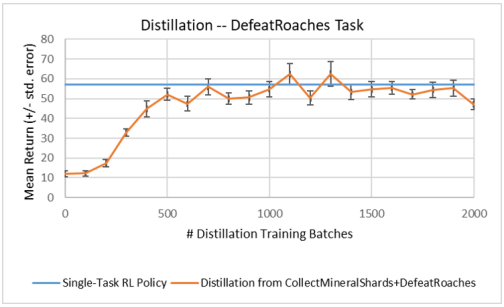
Zhang, D., Sun, Y., Eriksson, B., and Balzano, L. Deep unsupervised clustering using mixture of autoencoders. *arXiv preprint arXiv:1712.07788*, 2017.

A. Policy Distillation Results

We use a policy distillation approach for policy consolidation. As a proof-of-concept, we conducted an experiment using policy distillation to combine two SC2 policies – for the CollectMineralShards and DefeatRoaches tasks – into a single policy. In the experiment, we use real observations from both tasks for distillation, rather than sampling one set of observations from a generative model. Figure 7 compares the performance of the distilled policy to the performance of single task policies. Distillation is able to preserve the control knowledge embodied in both policies while compressing them into a single policy, and about 100x fewer training batches are required for distillation than were required originally to learn the policies being distilled. Policy distillation thus provides an effective and efficient means of knowledge consolidation for our framework.



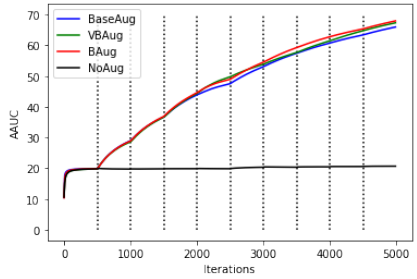
(a)



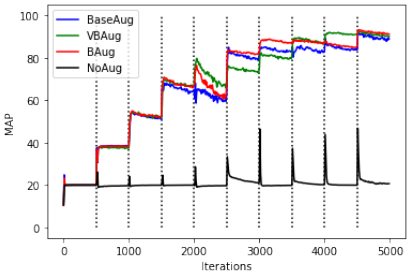
(b)

Figure 7. Policy distillation

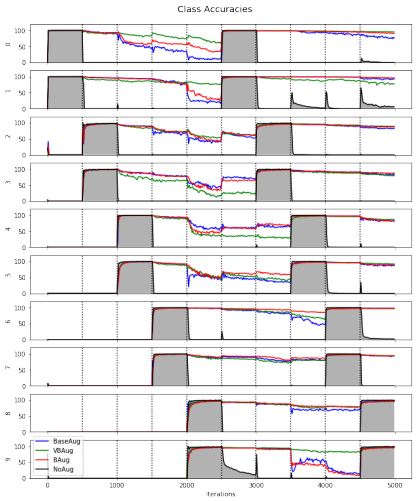
B. Supplementary Experimental Results



(a) Average area under the curve averaged over classes vs iterations.



(b) Mean average precision vs iterations.

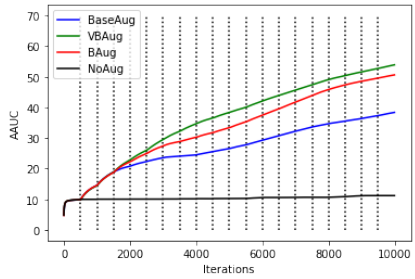


(c) Per-class (y-axis) accuracy over iterations (x-axis).

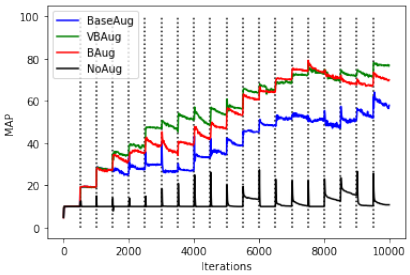
Figure 8. SplitMNIST: Per-task breakdown of accuracy vs iterations. Each task is seen for 500 iterations. Steady increase in these metrics indicates successfull continual learning without forgetting.

BuildMarines	CollectMineralShards	DefeatZerglingsAndBanelings	MoveToBeacon	CollectMineralsAndGas	DefeatRoaches
					

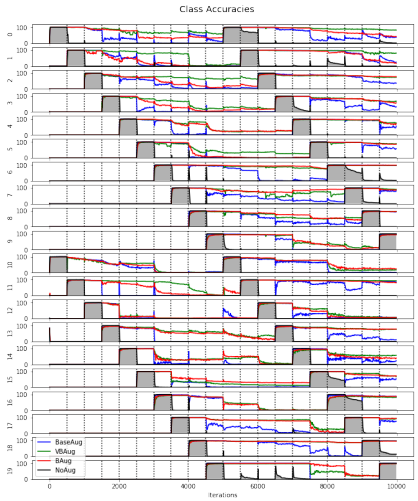
Table 3. SC2 Data Representation



(a) Average area under the curve averaged over classes vs iterations.



(b) Mean average precision vs iterations.



(c) Per-class (y-axis) accuracy over iterations (x-axis).

Figure 9. Split(MNIST+FashionMNIST): Per-task breakdown of accuracy vs iterations. Each task is seen for 500 iterations. Steady increase in these metrics indicates successfull continual learning without forgetting.

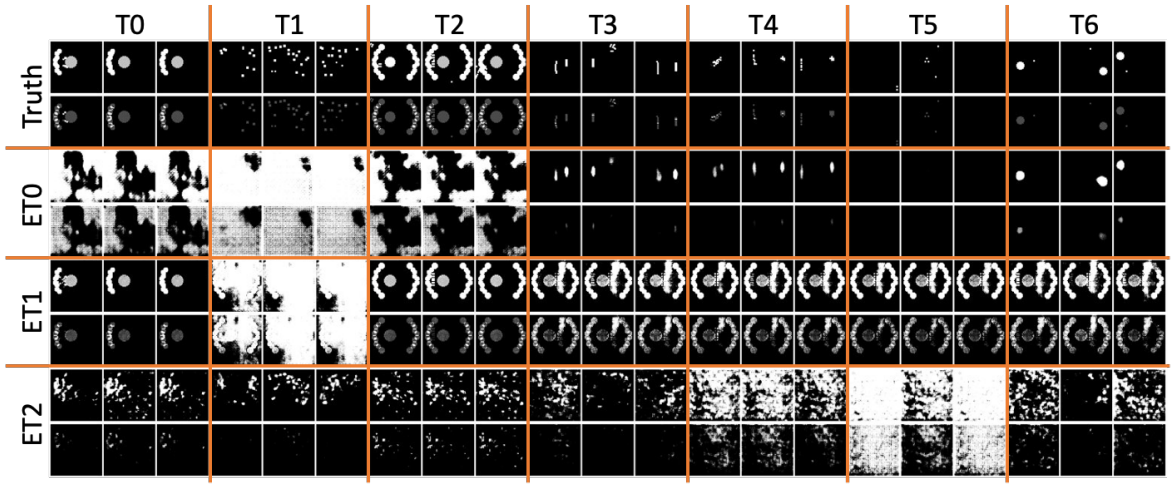


Figure 10. OWVAE reconstructions for SC2 tasks.