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

Anonymous Authors[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 with a generative model that can sample from the skill's input space. The model 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 performance comparable to the state-of-the-art in supervised continual learning benchmarks, and show encouraging preliminary results for forward knowledge transfer in a lifelong RL application in the video 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., 2015a;b; Tessler et al., 2016) among other areas. Recent work comprises many complementary approaches including knowledge consolidation using memory (van de Ven & Tolias, 2019), learning regularized parametric representations (Kirkpatrick et al., 2016; Huszár, 2018; Kirkpatrick et al., 2018), meta-learning (Nagabandi et al., 2019), neuromodulation (Masse et al., 2018), transfer of learned representation (Yosinski et al., 2014; Lee et al., 2019) and dynamic neural network architectures (Li et al., 2019; Rusu et al., 2016).

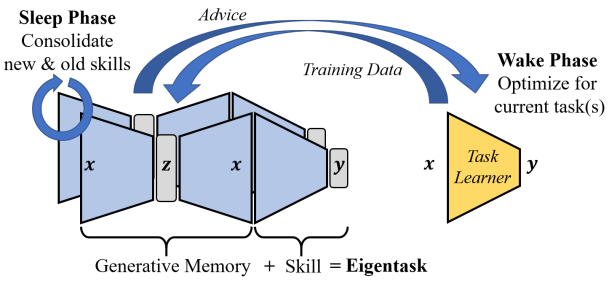


Figure 1. Our approach is based on *eigentasks*, which combine a skill with a generative model of its inputs. Our lifelong learning agents operate in a wake-sleep cycle, solving a stream of tasks during the wake phase, and consolidating new task knowledge during the sleep phase using generative replay.

This paper advances the state of the art in the generative replay approach to lifelong learning (van de Ven & Tolias, 2019; Raghavan et al., 2019; He, 2018; Ramapuram et al., 2017; 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 to replay old task data. Experimental evidence in biology shows that structured replay is one of the functions of mammalian sleep (Krishnan et al., 2019; Louie & Wilson, 2001; Wilson & McNaughton, 1994; Diekelmann & Born, 2010). 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

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

task – is an equally important lifelong learning problem that generative replay has not addressed. Furthermore, recent evidence in biology (McClelland et al., 2020) shows that only a few selected experiences are replayed during sleep, in contrast 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. Our model 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.

Our method partitions data into eigentasks using out-of-distribution detection based on a likelihood ratio test in the latent space of the generator (Ren et al., 2019). We present a new 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 eigentasks achieve performance comparable to state-of-the-art (SOTA) generative memory approaches RwF (van de Ven & Tolias, 2019) and DGR (Shin et al., 2017) in the standard splitMNIST benchmark for continual learning. We demonstrate that eigentasks achieve superior performance due to task disentanglement (Achille et al., 2018) on a new benchmark that contains a mix of multiple datasets.

Generative replay can have a deleterious effect if the generated samples are of low quality. 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 and create replay datasets that are balanced over labels. 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 consists of a sleep phase, where experience is consolidated into eigentasks, and a wake phase, where eigentasks are leveraged for rapid exploration and few-shot learning. In the wake phase, the eigentask memory generates advice by recalling options that are applicable in similar situations. 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 problem of solving a sequence of "mini-games" in the video game Starcraft 2 (SC2) (Vinyals et al., 2017). We demonstrate that our approach facilitates forward transfer to similar tasks. In most mini-games we observe a jump start in the accumulated reward, and in one mini-game, knowledge transfer allows our approach to outperform the single-task learner by 1.5x with 10x fewer samples.

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 thus we simplify the definition of a task to $T_i = (P_i, L_i)$. We denote the (potentially infinite) set of tasks by $\mathcal{T}$. Since a task T_i is totally characterized by its index i, we sometimes use tasks and their indices interchangeably.

The task sequence seen by the agent 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. Let $\mathbf{X}^n$ denote a bag of n instances $X^{(1)}, \ldots, X^{(n)}$ where each $X^{(j)} \sim P$ for some input distribution P. 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,

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

Equation 1 defines the optimal hypothesis for a particular task distribution. Note that the task distribution W is unknown to the agent, and $\mathbf{X}^n$ does not contain the task index. Due to the unknown task distribution, Eq. 1 is hard to optimize from data. Lifelong learning is strictly harder than multi-task learning, where instances are paired with task IDs, thus the outer expectation is a simple average.

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

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

Note that this definition removes the factor of n_i from the per-task expected loss, and thus weights all tasks equally regardless of how often or for how long they appear.

Backward Transfer (BT) describes the difference in performance on familiar tasks before and after one or more epochs of learning. We define BT after epoch j as the difference in loss before and after learning epoch j restricted to tasks seen in previous epochs. 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 given by,

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

Negative BT corresponds to forgetting knowledge learned in previous episodes, and positive BT could indicate successful knowledge consolidation between related tasks.

Forward Transfer (FT) describes the difference in learning performance on novel tasks with and without training on previous tasks. We define one-step FT as the difference in loss on the novel tasks in a learning epoch when learning from scratch versus when learning with prior knowledge from previous epochs,

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

The differences within parentheses restrict the loss to novel tasks in the jth episode. Positive FT indicates transfer of knowledge or skills between related tasks, and typically corresponds to a jump start in performance and lower sample complexity to achieve a good solution.

One can also consider expected BT/FT with respect to the task distribution W, which influences the expectation via the training data consumed in updating the hypothesis $h^{1:j-1} \to h^{1:j}$.

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 $(\mathcal{S}, \mathcal{A}, P, R, P_0, \gamma)$, 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, P_0 is the distribution of initial states, and $\gamma \in (0, 1]$ is the discount factor. The objective is to find a policy $\pi : \mathcal{S} \mapsto \text{Prob}(\mathcal{A})$ that maximizes the value function $V^\pi(s) = E_{S_t \sim P, A_t \sim \pi}[\sum_{t=1}^{\infty} \gamma^t R(S_t, A_t)]$. Intuitively, a task in lifelong RL is an MDP, where all share the same state and action spaces, but P, R, and P_0 differ. The input space $\mathcal{X}$ is the set of states $\mathcal{S}$, the output space $\mathcal{Y}$ is the set of actions $\mathcal{A}$, and the loss function is the negative of the return, $L(\mathbf{x}, \mathbf{y}) = -\sum_{(x,y) \in (\mathbf{x}, \mathbf{y})} R(x, y)$. The input distribution is the distribution of state trajectories induced by the agent's behavior policy.

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. Generative replay addresses the scalability of this technique by representing the data compactly with a generative model. We define a generative memory as 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.

Generative memory (GM) can be used for continual learning by augmenting the training data of a new episode $\{X_{real}, Y_{real}\}$ with a generated dataset $\{X_{mem}, Y_{mem}\}$, and optimizing the learner on the union of real and recalled datasets. With perfect recall and sufficient samples, GM can achieve continual learning without negative BT. In practice, the recall deteriorates due to the fixed capacity of the generator and the underlying hard optimization problem, causing some negative BT as the number of tasks grows. Prior works on generative memory (van de Ven & Tolias, 2019; He, 2018; Ramapuram et al., 2017; Shin et al., 2017) can be considered minor variations on GM. This approach has the fundamental limitation that selective knowledge transfer is not possible because all previous tasks are replayed.

We aim to leverage selective GM for positive forward transfer, in addition to avoiding negative BT. To achieve this, 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 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) cluster inputs based on "perceptual similarity" alone, which can result in overlap in skills. On the other hand, mixtures-of-experts (Tsuda et al., 2020) capture

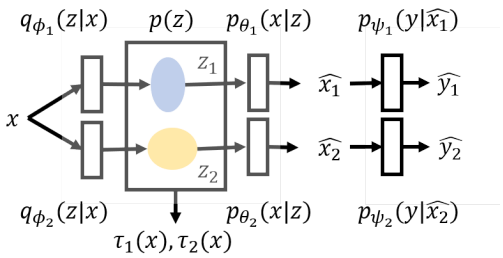


Figure 2. An Open World Auto Encoder (OWVAE) with two eigentasks, showing the terms used in the loss function (Eq. 9).

"skill similarity" alone. Our key idea is to combine the MoE concept with generative replay, to avoid catastrophic forgetting and realize selective transfer for few-shot learning while learning the experts incrementally.

3.1. Eigentask Definition

Formally, an eigentask is a tuple $E = (g, f)$ comprising a generator $g(\epsilon)$ that defines a probability distribution on the input set $\mathcal{X}$ 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 m eigentasks. Typically, g, f and τ are parameterized functions such as DNNs. Intuitively, τ measures the degree to which each eigentask "fits" the current input. 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,

$$\min E_{i\sim\tau}[\mathcal{L}_{gen}(g_i(\epsilon)|X) + \mathcal{L}_{disc}(f_i|g_i(\epsilon), X, Y)] \quad (5)$$

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.

In this work, we use a VAE (Kingma & Welling, 2014) as the generator, with latent space denoted by z, encoder $q_\phi(z|x)$, decoder $p_\theta(x|z)$ and prior $z \sim N(0, \mathbf{I})$. We use reconstruction error between the input and decoder output as the generative loss $\mathcal{L}_{gen}$. We propose to use likelihood ratios to define $\tau(x)$ as in (Ren et al., 2019),

$$z_i \sim q_{\phi_i}(z|x) \quad (6)$$

$$\tau_i(x) = \text{softmax}_\beta\left(\frac{p_{\theta_i}(x|z)}{\max_j p_{\theta_j}(x|z)}\right) \quad (7)$$

$$\approx \text{softmax}_\beta\left(\frac{\Phi(z_i)}{\max_i \Phi_i(z_i)}\right) \quad (8)$$

where subscript i denotes eigentask i (Figure 2), Eq. 7 uses the probability of decoder i generating the observed data x for some z, and Eq. 8 assumes that decoder weights are the inverse of the encoder weights, so the likelihood is defined in terms of the density of the standard Gaussian Φ applied to the encoding z_i. The parameter β is the softmax temperature. Intuitively, among the encodings z_i, we should pick the one that has a higher prior likelihood. Finally, we take the softmax to get a probability distribution over eigentasks for each instance x, and calculate the loss according to Eq. 5 using $\tau(x)$ as instance weights. The effect of the likelihood ratio weights it to cause only one eigentask to "claim" each sample, thus partitioning the data. We refer to this model as an Open World Variational Auto Encoder (OWVAE). Figure 2 shows an OWVAE with two eigentasks. The overall loss function for OWVAE is in Eq. 9.

$$\min_{\theta,\phi,\psi} E_{x,y}[E_\tau[\mathcal{L}_{VAE}(x;\theta,\phi) + \log p_\psi(y|\hat{x})] + \lambda \sum_{i,j} |\cos(z_i, z_j)|] \quad (9)$$

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

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

The last term encourages separation between the encodings of eigentasks for the same input. 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).

3.2. The Wake-Sleep Cycle

In our approach, we convert the streaming input to a streaming-batch setting by dividing the stream into a sequence of learning epochs (Section 2.1). We view this in biological terms as dividing the agent's lifetime into two distinct phases: a *wake phase* and a *sleep phase*.

In the wake phase, the learner's goal is to act correctly on incoming inputs. In supervised learning that might mean simply outputting the correct label, while in reinforcement learning the agent needs to explore the new tasks. During the wake phase, the learner stores incoming 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 describe instantiations of the wake-sleep cycle for supervised learning and RL in the next two sections.

4. Lifelong Supervised Learning using Eigentasks

In lifelong SL, the wake phase is fairly trivial. The agent should simply label the incoming examples according to its current model. In the eigentask framework, this means feeding incoming examples x into each eigentask generator, sampling an eigentask $i \sim \tau(x)$, and outputting the label assigned by the skill f_i, which will tend to be a skill that was applicable to previous similar examples. The eigentask model could be leveraged in other ways during the wake phase, for example using selective replay to create augmented training data to help learn the current examples quickly, but we have not yet investigated these possibilities.

The sleep phase is more interesting. We assume that a batch of labeled examples X_{new}, Y_{new} are available from the wake phase. We build on typical approaches to continual learning using generative replay (van de Ven & Tolias, 2018; Achille et al., 2018). First, a copy of the OWVAE model is created and used to generate batches of replay. A batch of random latent vectors z is generated for each eigentask and forward-propagated through the decoder and skill to generate examples of the form $(X_i, Y_i, \text{conf}_i)$, $i = 1, \ldots, m$, where conf_i denotes the confidence of the prediction Y_i output by skill f_i. Second, the OWVAE runs multiple iterations of gradient-based learning on the batch $(X_{new}, Y_{new}) \cup \bigcup_{i=1}^{n} (X_i, Y_i)$, where each iteration uses a randomly generated batch from the OWVAE copy. Finally, the copy is disposed and consolidation is complete.

In the batch-streaming regime, especially early in training, there will be periods where some eigentasks have received little or no training; for example, τ may give low weight to some eigentasks by chance. These untrained generators will tend to generate meaningless noise that would be harmful if incorporated in the training set. To mitigate this problem, we developed a rejection sampling strategy leveraging the confidence of the skill associated with the generator. We reject a sample (x, y) from generator g_i if the confidence of the associated skill is below a threshold, $\text{conf}_i(x, y) < \delta$. We also use the skill to self-label the generated examples, which enables us to generate label-balanced data by rejecting over-represented labels. These two refinements to the sampling process were critical in achieving SOTA on a lifelong SL benchmarks (Section 6.2).

5. Lifelong RL using Eigentasks

In the lifelong RL setting, our tasks are MDPs (Section 2.2), the eigentask skills f_i are MDP policies, and the associated generators g_i generate observations from the portions of the state space where the policies should be applied.

5.1. Wake Phase

In the wake phase, we leverage knowledge of previous tasks to solve the current task more quickly. One of the key determinants of RL performance is the efficiency of exploration. Often, the hardest part is finding the goal for the first time. Without any task knowledge, RL techniques typically resort to random exploration in the early stages of learning.

Our approach to knowledge transfer in the wake phase is to use eigentask skills to aid in exploration. Transfer is accomplished using off-policy training with a behavior policy that 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 f_i be the ith skill. Given an observation o, the behavior policy $b(o)$ selects an action according to

$$b(o) = \begin{cases} \pi(o) & \text{w.p. } 1 - \mu(o,t) \\ f_i(o) & \text{w.p. } \mu(o,t)\tau(o)_i \end{cases}$$

where $\mu(o, t) \in [0, 1]$ is a "mixing" function that optionally decays over time. We use the off-policy actor-critic algorithm VTrace (Espeholt et al., 2018) in our experiments, but our approach is compatible with any off-policy algorithm.

5.2. Sleep Phase

In the sleep phase, our goal is to consolidate the control knowledge in the target policy π into the eigentask skills f_i. Our approach is based on policy distillation (Rusu et al., 2015). During training on the target task, we collect a set of observations O_T, and at the conclusion of the wake phase we compute the target policy's output for observation to form a training dataset $D_T = \{(o, \pi(o)) : o \in O_T\}$. This transforms the problem of imitating the target policy to a SL problem. The sleep phase proceeds in the same manner as for supervised learning tasks (Section 4). The new training examples in D_T are fed to the OWVAE model, which allocates them to its different branches according to the τ function. The new examples are augmented with generative replay datasets $D_i = \{(o, f_i(o) : o \sim g_i\}$ for each eigentask $E_i = (g_i, f_i)$.

6. Experiments

We validate eigentasks on unsupervised, supervised classification and RL tasks. We show experimentally that eigentasks achieve performance comparable to state-of-the-art generative memory approaches in the standard splitMNSIT

benchmark for continual learning. We demonstrate that eigentasks achieve superior performance on a new benchmark that contains a mix of multiple datasets. We demonstrate our lifelong RL algorithm on the Starcraft 2 (SC2) (Vinyals et al., 2017) mini-games benchmark. We demonstrate that our approach compares favorably to the baselines of single-task and multi-task learning.

6.1. Illustration: synthetic problem

In this section, we illustrate the OWVAE model on a synthetic but challenging problem for current generative memory approaches. The problem is inspired by the Wisconsin Card Sorting task (Tsuda et al., 2020). The tasks cannot be distinguished by "perceptual similarity" but can be distinguished by "skill similarity", so an MoE model can solve the tasks (Tsuda et al., 2020).

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. We test to see if an OWVAE with two eigentasks can separate the two tasks and achieve a higher accuracy. Figure 3b shows that indeed the accuracy is higher (> 0.7) when using two eigentasks. 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 and vice-versa. In contrast, current generative replay methods will not work in this problem because learning either task causes forgetting of the other.

6.2. Continual Learning for Supervised Classification

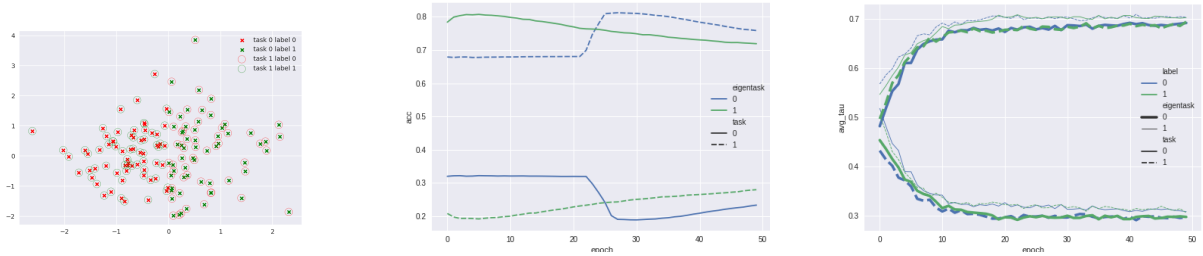
We use the class-incremental learning (Class-IL) setting introduced in (van de Ven & Tolias, 2019). In this setting, new classes or groups of classes are presented incrementally to the learning alorithm. We demonstrate continual learning without forgetting on the standard splitMNIST problem and compare to the state-of-the-art. 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 and refer to it as split(M+F). The combined MNIST and Fashion-MNIST is split into ten tasks. Each task introduces two new classes, one MNIST digit and one fashion article. We demonstrate that eigentasks can separate the datasets and tasks while learning each task without forgetting. In both benchmarks, each task is seen for 500 iterations with a batch size of 32. All compared approaches use a two layer perceptron with 400 units for the encoder and decoder, and a latent dimension of 100.

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+distill	91.79	72.40
	RtF	92.56	61.15
Replay+Exemplars	iCaRL	94.57	82.69
Replay (ET)	ET1-BaseAug	87.68	69.29
	ET1-BAug	90.99	74.11
	ET1-VBAug	90.69	77.43
	ET2-BaseAug	88.93	57.91
	ET2-BAug	**91.27**	69.95
	ET2-VBAug	90.25	**76.81**

Table 1. Average test accuracy over all tasks on splitMNIST (D1) and split(MNIST+FashionMNIST) (D2) tasks. ET1 and ET2 denote the number of eigentasks in an OWVAE model. The comparison methods are 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).

We compare our approach against several continual learning approaches spanning regularization, replay, and replay with exemplars. We include two baselines of single-task learner (lower bound - only learns on current task) and offline multi-task learner (upper bound - knows all tasks). To study the impact of our rejection sampling strategy (Section 4), 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, and (3) VBAug: rejecting low confidence examples and creating label-balanced replay. The inputs to the skill are the mid-level features of the corresponding VAE, i.e., the activations in the last layer of the encoder. Training is done on the the standard train sets and results are reported on the standard test sets. The average accuracies over all tasks at the end of learning 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 comparable to the single task learner, i.e., they learn and immediately forget each task. Our approach (Replay+Eigentasks) has accuracy comparable to other replay-based methods (except LwF) on splitM. However, on split(M+F), our approach has a higher accuracy (about 4% higher). Unsurprisingly, using exemplars within replay improves accuracy further. Exemplars could be integrated with eigentasks in future work.

A detailed breakdown of accuracy per-task over time is shown in the appendix. Figure 4 shows the task separation by visualizing reconstructions for each task. It shows that



(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.

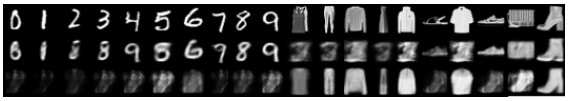


Figure 4. Split(MNIST+FashionMNIST): Image reconstruction by OWVAE(2) showing task separation. Top row: Ground truth. Middle and Bottom: reconstruction by first and second eigentask.

the first eigentask has learned all the MNIST digits and the second eigentask has learned all the fashion articles. The first eigentask has also learned some fashion articles whereas the second eigentask has not learned any digits.

6.3. Starcraft 2

We use the Starcraft 2 learning environment (SC2) (Vinyals et al., 2017) as a setting for lifelong RL. SC2 is a rich and complex platform in which many diverse RL tasks can be implemented. We used their "mini-games" as the task set for our experiments. Our agent architecture is a slight modification of their FullyConv architecture.

6.3.1. Eigentask Learning

We use the feature maps extracted by SC2LE instead of the RGB input. SC2LE extracts 17 feature maps of size 64x64. We restrict our experiments to use two key features 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. We 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(3) incrementally in a fully unsupervised manner, within a continual learning experiment similar to splitMNIST (each mini-game is seen for 500 iterations). SC2 mini-games are perceptually different, so the OWVAE was able to separate the tasks in to meaningful eigentasks. As shown in Figure 5, the first eigentask correctly grouped all combat tasks together, but incorrectly grouped a navigation task due to the unsupervised training. The second eigentask learned the BuildMarines task alone, whereas the third eigentask grouped the resource gathering tasks together. Figure 5 also shows the variation in τ over iterations. 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 reconstructions shown in appendix.

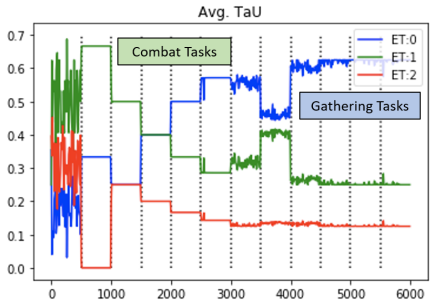
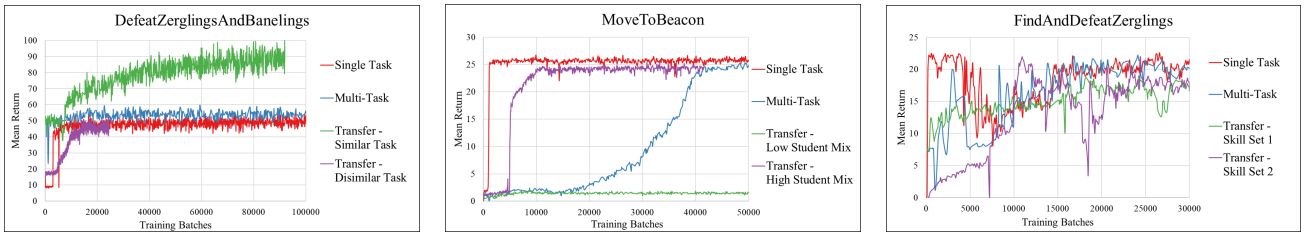


Figure 5. Continual Unsupervised Learning of a OWVAE(3) in SC2 mini-games: Variation of τ over iterations and learned task grouping. Each mini-game is observed for 500 iterations.

6.3.2. Forward Policy Transfer

These experiments examine forward policy transfer from previously-learned tasks. We first trained the generative part of a two-branch OWVAE in an unsupervised manner on observations collected from random policies. Each experiment looks at transfer from two source policies – each paired with one of the OWVAE branches – to a target policy being learned for a new task. The source policies are selected from trained single-task policies for one of the tasks that the OWVAE has clustered into the corresponding branch, determined by manual inspection. Training uses the VTrace learning rule (Espeholt et al., 2018) in a concurrent A2C implementation heavily adapted from open-source code published by Ring (2018). We examined transfer from source policies trained on source tasks that are either similar or dissimilar to the target task (Table 2). Our main results with forward transfer are summarized in Figure 6. All the plots



(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.

include two baselines: the performance of a Single Task policy trained on the target task, and the performance of a Multitask policy trained on batches containing experience from all 6 tasks. The vertical axis shows mean per-episode return obtained while following the behavior policy during training.

Experiment (6a) examines transfer to the DefeatZerglingsAndBanelings task. In the Transfer-Similar Task condition, the source task policies are 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 source policies are CollectMineralShards and MoveToBeacon. Transfer in the Similar Task condition resulted in both good performance from the start of training and substantially better ultimate performance compared to Single Task training. Notably, the ultimate performance with forward transfer is also better than the best published performance for the FullyConv policy architecture (Vinyals et al., 2017). Transfer in the Dissimilar Task condition still resulted in better initial performance than Single Task training, but ultimately slowed down learning. However, the performance still improved and became comparable to the Single Task policy.

In experiment (6b), we study transfer to the MoveToBeacon task. Because of the way we trained the OWVAE, the MoveToBeacon task gets clustered with the two Combat tasks, and not with the more similar CollectMineralShards task. This is because unsupervised training of the OWVAE ignores dynamics and rewards. As a result, the OWVAE τ function selects a combat policy as the source policy for transfer to MoveToBeacon. When we use the default schedule for the behavior policy mixing rate $\mu(o, t)$, the dissimilarity of the source tasks causes transfer to hinder learning. However, using a different mixing schedule that gives more weight to the target policy allows the agent to overcome the effect of task disimilarity and still learn the target task. Thus, the mixing rate μ gives us a mechanism to modulate the strength of knowledge transfer according to task similarity.

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 source task sets – either CollectMineralShards+DefeatRoaches or CollectMineralShards+MoveToBeacon – but transfer had no clear positive or negative effect.

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 or recurring 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 transfer of knowledge. Our rejection sampling trick using eigentask skills allowed us to achieve SOTA performance on continual learning benchmarks. In RL, we demonstrated successful forward transfer of policies to novel tasks in a lifelong RL problem in Starcraft 2, and exceeded the best published performance for our policy architecture on one of the Starcraft 2 tasks due to the use of knowledge transfer.

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 proposed 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 discriminate RL tasks that have similar observations but different dynamics.

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*, 2015a.

Ammar, H. B., Tutunov, R., and Eaton, E. Safe Policy Search for Lifelong Reinforcement Learning with Sublinear Regret. In *ICML*, 2015b.

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.

Raghavan, A., Hostetler, J., and Chai, S. Generative Memory for Lifelong Reinforcement Learning. *arXiv preprint arXiv:1902.08349*, 2019.

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

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.

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.

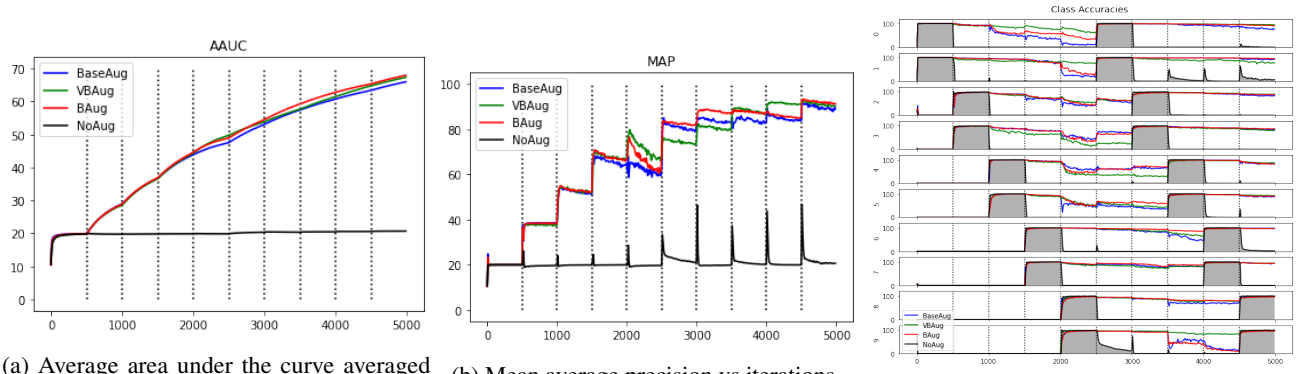
Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, 2014.

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. Lifelong Learning using Eigentasks: Task Separation, Skill Acquisition and Selective Transfer with Application to Reinforcement Learning in Starcraft 2 — Supplementary Experimental Results

A.1. Breakdown of results for continual learning

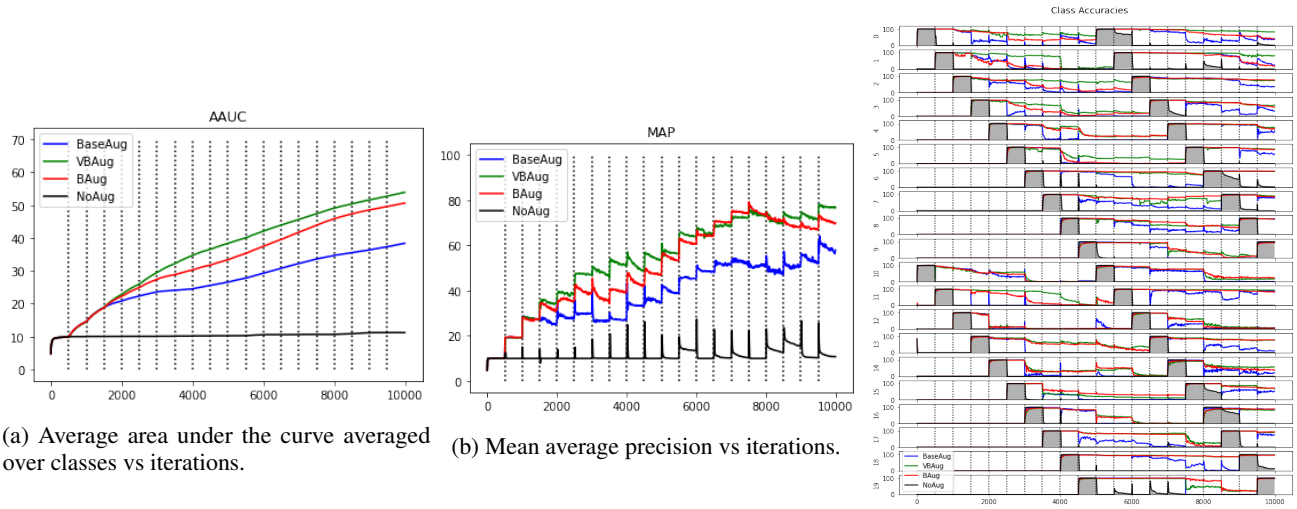


(a) Average area under the curve averaged over classes vs iterations.

(b) Mean average precision vs iterations.

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

Figure 7. 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.



(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. 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.

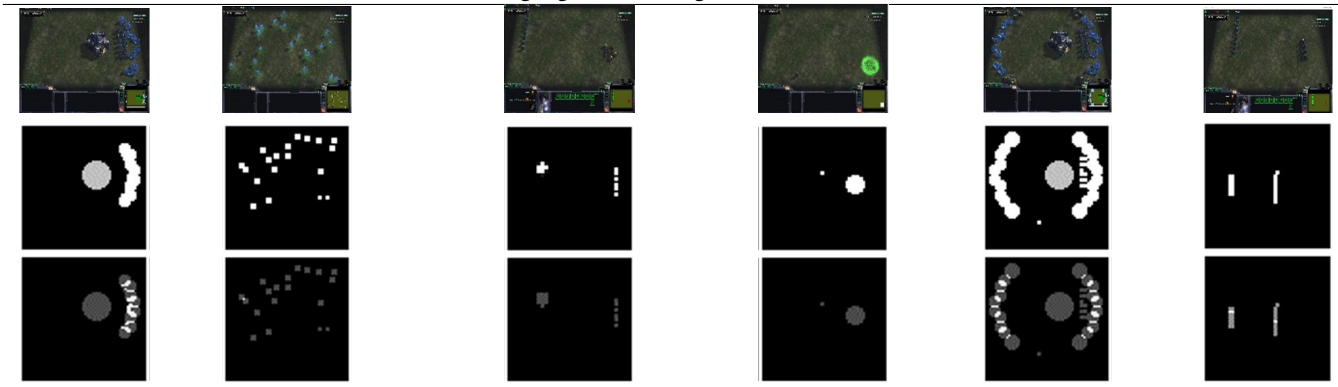
BuildMarines	CollectMineralShards	DefeatZerglingsAndBanelings	MoveToBeacon	CollectMineralsAndGas	DefeatRoaches
					

Table 3. SC2 Data Representation

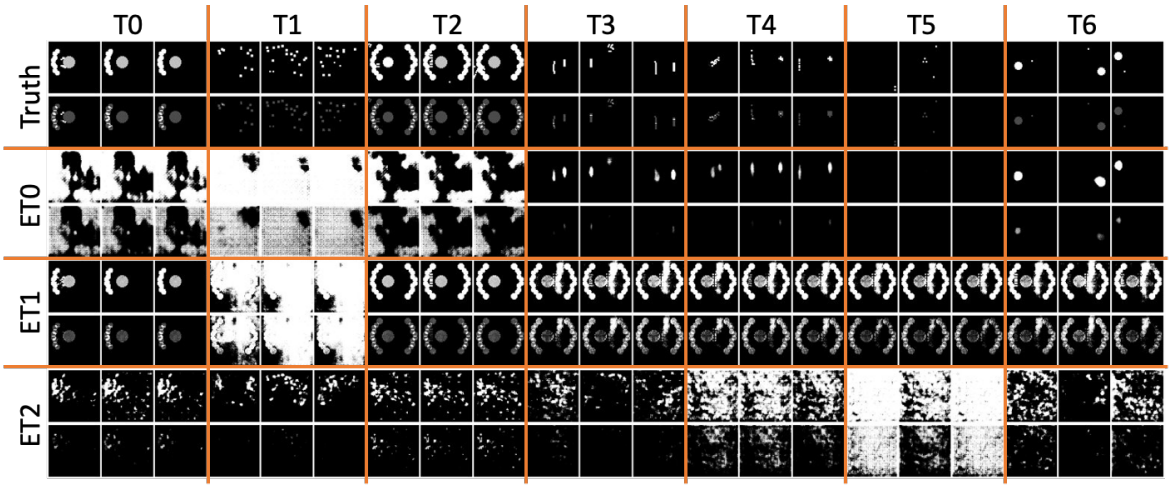
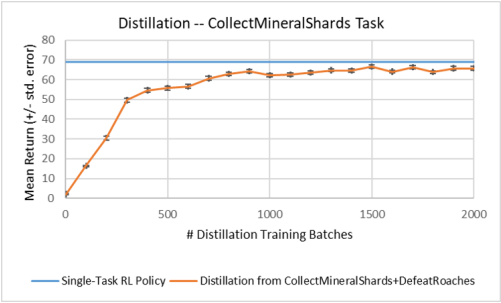


Figure 9. Model reconstructions for SC2 tasks.

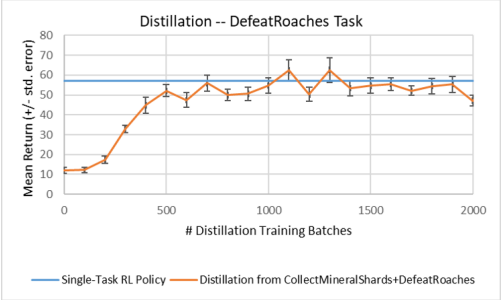
A.2. Policy Distillation Results

A.2.1. Policy Consolidation

We use a policy distillation approach (Rusu et al., 2015) for policy consolidation (Section **??**). 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 10 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 10. Policy distillation