Journal of Artificial Intelligence Research 67 (2020) 673-703 Submitted 01/2019; published 03/2020

Using Task Descriptions in Lifelong Machine Learning for Improved Performance and Zero-Shot Transfer

Mohammad Rostami MROSTAMI@SEAS.UPENN.EDU
David Isele ISELE@SEAS.UPENN.EDU
Eric Eaton EEATON@SEAS.UPENN.EDU
University of Pennsylvania,
Philadelphia, PA 19104 USA

Abstract

Knowledge transfer between tasks can improve the performance of learned models, but requires an accurate estimate of inter-task relationships to identify the relevant knowledge to transfer. These inter-task relationships are typically estimated based on training data for each task, which is inefficient in lifelong learning settings where the goal is to learn each consecutive task rapidly from as little data as possible. To reduce this burden, we develop a lifelong learning method based on coupled dictionary learning that utilizes high-level task descriptions to model inter-task relationships. We show that using task descriptors improves the performance of the learned task policies, providing both theoretical justification for the benefit and empirical demonstration of the improvement across a variety of learning problems. Given only the descriptor for a new task, the lifelong learner is also able to accurately predict a model for the new task through zero-shot learning using the coupled dictionary, eliminating the need to gather training data before addressing the task.

1. Introduction

Transfer learning (TL) and multi-task learning (MTL) methods reduce the amount of experience needed to train individual task models by reusing knowledge from other related tasks. This transferred knowledge can improve the training speed and model performance, as compared to learning the tasks in isolation following the classical machine learning pipeline (Pan & Yang, 2010). TL and MTL techniques typically select the relevant knowledge to transfer by modeling inter-task relationships using a shared representation, based on training data for each task (Baxter, 2000; Ando & Zhang, 2005; Bickel, Sawade, & Scheffer, 2009; Maurer, Pontil, & Romera-Paredes, 2013). Despite benefits over single-task learning, this process requires sufficient training data for each task to identify these relationships before knowledge transfer can succeed and improve generalization performance. This need for data is especially problematic in learning systems that are expected to rapidly learn new tasks during real-time interaction with the environment: when faced with a new task, the learner would first need to gather data on the new task before bootstrapping a model via transfer, consequently delaying how quickly the learner could address the new task.

Consider instead the human ability to rapidly bootstrap a model for a new task given *only* a *high-level task description*—before obtaining experience on the actual task. For example, viewing only the image on the box of a new IKEA chair, we can immediately identify

[†]An earlier version of this work focusing on policy gradient reinforcement learning appeared in the proceedings of IJCAI 2016 (Isele, Rostami, & Eaton, 2016).

related previous assembly tasks and begin formulating a plan to assemble the chair. In the same manner, an experienced inverted pole balancing agent may be able to predict the controller for a new pole given its mass and length, prior to interacting with the physical system. These examples suggest that an agent could similarly use high-level task information to bootstrap a model for a new task more efficiently.

Inspired by this idea, we explore the use of high-level task descriptions to improve knowledge transfer between multiple machine learning tasks from a single domain. We focus on lifelong learning scenarios (Thrun, 1996; Ruvolo & Eaton, 2013; Chen & Liu, 2018; Rostami, Kolouri, & Pilly, 2019), in which multiple tasks arrive consecutively and the goal is to rapidly learn each new task by building upon previous knowledge. Our approach to integrating task descriptors into lifelong machine learning is general, as demonstrated on applications to reinforcement learning, regression, and classification problems. In reinforcement learning settings, our idea can be compared with the universal value function approximation algorithm by Schaul et al. (2015) in that the goal is to generalize the learned knowledge to other unexplored scenarios. Schaul et al. incorporate the goals of an RL learner into the value function to allow for generalization over unexplored goals. In contrast, our goal is to learn a mapping from high-level task descriptions onto the optimal task parameters, enabling the agent to learn future tasks without exploration solely using high-level descriptions of those tasks.

Our algorithm, Task Descriptors for Lifelong Learning (TaDeLL), encodes task descriptions as feature vectors that identify each task, treating these descriptors as side information to augment training data on the individual tasks. The idea of using task features for knowledge transfer has been explored previously by Bonilla et al. (2007) in an offline batch MTL setting[1], and more recently by Sinapov et al. (2015) in a computationally expensive method for estimating transfer relationships between pairs of tasks. Svetlik et al. (2017) also use task descriptors to generate a curriculum that improves the learning performance in the target task by learning the optimal order in which tasks should be learned. In comparison, our approach operates online over consecutive tasks in a setting where the agent does not control the order in which tasks are learned.

We use *coupled dictionary learning* to model the inter-task relationships between the task descriptions and the individual task models or policies in lifelong learning. The coupled dictionary enforces the notion that tasks with similar descriptions should have similar models or policies, but still allows dictionary elements the freedom to accurately represent the different task models or policies. We connect the coupled dictionaries to the PAC-learning framework, providing theoretical justification for why the task descriptors improve performance, and verify this improvement empirically.

In addition to improving the task models, we show that the task descriptors enable the learner to accurately predict the models or policies for unseen tasks given only their description—this process of learning without data on future tasks is known as *zero-shot learning*. This capability is particularly important in the online setting of lifelong learning. It enables the system to accurately predict policies for new tasks through transfer from

[1]Note that "batch learning" in this context refers to offline learning, when all tasks are available a priori and is not related to the notion of a "batch" of data in first-order optimization methods such as mini-batch stochastic gradient descent.

previously learned tasks, without requiring the system to pause to gather training data on each future task.

Specifically, this article provides the following contributions:

- We develop a general mechanism based on **coupled dictionary learning** to incorporate task descriptors into knowledge transfer algorithms that use a factorized representation of the learned knowledge to facilitate transfer (Kumar & Daumé, 2012; Maurer et al., 2013; Ruvolo & Eaton, 2013; Kolouri, Rostami, Owechko, & Kim, 2018).
- Using this mechanism we develop **two algorithms**, for lifelong learning (TaDeLL) and MTL (TaDeMTL), that incorporate task descriptors to improve learning performance.
- Most critically, we show how these algorithms can achieve **zero-shot transfer** to bootstrap a model for a novel task, given only the high-level task descriptor.
- We provide **theoretical justification** for the benefit of using task descriptors in lifelong learning and MTL, building on the PAC-learnability of the framework.
- Finally, we demonstrate the empirical effectiveness of TaDeLL and TaDeMTL on **reinforcement learning** scenarios involving the control of dynamical systems, and on prediction tasks in **classification** and **regression** settings, showing the generality of our approach.

2. Related Work

Multi-task learning (MTL) (Caruana, 1997) methods often model the relationships between tasks to identify similarities between their datasets or underlying models. There are many different approaches to modeling these task relationships. Bayesian approaches take a variety of forms, making use of common priors (Wilson, Fern, Ray, & Tadepalli, 2007; Lazaric & Ghavamzadeh, 2010), using regularization terms that couple task parameters (Evgeniou & Pontil, 2004; Zhong & Kwok, 2012), and finding mixtures of experts that can be shared across tasks (Bakker & Heskes, 2003).

Whereas Bayesian MTL methods aim to find an appropriate bias to share among all task models, transformation methods seek to make one dataset look like another, often in a transfer learning setting. This can be accomplished with distribution matching (Bickel et al., 2009), inter-task mapping (Taylor, Stone, & Liu, 2007), or manifold alignment techniques (Wang & Mahadevan, 2009; Ham, Lee, & Saul, 2005).

Both the Bayesian strategy of discovering biases and the shared spaces often used in transformation techniques are implicitly connected to methods that learn shared knowledge representations for MTL. For example, the original MTL framework developed by Caruana (1997) and later variations (Baxter, 2000) capture task relationships by sharing hidden nodes in neural networks that are trained on multiple tasks. Alternatively, the distributions of several tasks can be related through a hidden layer in neural networks (Long, Zhu, Wang, & Jordan, 2017; Rostami, Kolouri, Eaton, & Kim, 2019). Related work in dictionary learning techniques for MTL (Maurer et al., 2013; Kumar & Daumé, 2012) factorize the learned models into a shared latent dictionary over the model space to facilitate transfer. Individual task models are then captured as sparse representations over this dictionary; the task relationships are captured in these sparse codes.

The Efficient Lifelong Learning Algorithm (ELLA) framework (Ruvolo & Eaton, 2013) used this same approach of a shared latent dictionary, trained online, to facilitate transfer as tasks arrive consecutively. The ELLA framework was first created for regression and classification (Ruvolo & Eaton, 2013), and later developed for policy gradient reinforcement learning (PG-ELLA) (Bou Ammar, Eaton, & Ruvolo, 2014) and collective multi-agent learning (Rostami, Kolouri, Kim, & Eaton, 2018) using distributed optimization (Hao, Oghbaee, Rostami, Derbinsky, & Bento, 2016). Other approaches that extend MTL to online settings also exist (Cavallanti, Cesa-Bianchi, & Gentile, 2010). Saha et al. (2011) use a task interaction matrix to model task relations online and Dekel et al. (2006) propose a shared global loss function that can be minimized as tasks arrive.

However, *all* these methods use task data to characterize the task relationships—this explicitly requires training on some data from each task in order to perform transfer. Instead of relying solely on the tasks' training data, several works have explored the use of high-level task descriptors to model the inter-task relationships in MTL and transfer learning settings. Task descriptors have been used in combination with neural networks (Bakker & Heskes, 2003) to define a task-specific prior and to control the gating network between individual task clusters. Bonilla et al. (2007) explore similar techniques for multi-task kernel machines, using task features in combination with the data for a gating network over individual task experts to augment the original task training data. These papers focus on multi-task classification and regression in batch settings where the system has access to the data and features for all tasks, in contrast to our study of task descriptors for lifelong learning over consecutive tasks. We use coupled dictionary learning to link the task description space with the task model's or task policy's parameter space. This idea was originally used in image processing (Yang, Wright, Huang, & Ma, 2010) and was recently explored in the machine learning literature (Xu, Hospedales, & Gong, 2016). The core idea is that two feature spaces can be linked through two dictionaries that are coupled by a joint sparse representation.

Several works consider transferring knowledge from a single source task to a single target task (Da Silva & Costa, 2017; Song, Gao, & Wang, 2018; Silva & Costa, 2018) using task descriptors. We focus on a lifelong learning framework, where the goal is to transfer knowledge from several previously learned tasks. In the work most similar to our problem setting, Sinapov et al. (2015) use task descriptors to estimate the transferability between each pair of tasks for transfer learning. Given the descriptor for a new task, they identify the source task with the highest predicted transferability, and use that source task as a warm start in reinforcement learning (RL). Though effective, their approach is computationally expensive, since they estimate the transferability for every task pair through repeated simulation. Their evaluation is also limited to a transfer learning setting, and they do not consider the effects of transfer over consecutive tasks or updates to the transferability model, as we do in the lifelong setting.

Our work is also related to zero-shot learning, which seeks to successfully label out-of-distribution examples, often through means of learning an underlying representation that extends to new tasks and uses outside information that appropriately maps to the latent space (Palatucci, Hinton, Pomerleau, & Mitchell, 2009; Socher, Ganjoo, Manning, & Ng, 2013; Rostami, Kolouri, McClelland, & Pilly, 2020). The ESZSL method by Romera-Paredes and Torr (2015) also uses task descriptions. Their method learns a multi-class

linear model, and factorizes the linear model parameters, assuming the descriptors are coefficients over a latent basis to reconstruct the models. Our approach assumes a more flexible relationship: that both the model parameters and task descriptors can be reconstructed from separate latent bases that are coupled together through their coefficients. In comparison to our lifelong learning approach, the ESZSL method operates in an offline multi-class setting.

3. Background

Our proposed framework for lifelong learning with task descriptors supports both supervised learning (classification and regression) and reinforcement learning settings. For completeness, we briefly review these learning paradigms here.

3.1 Supervised Learning

Consider a standard batch supervised learning setting. Let $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ be a d-dimensional vector representing a single data instance with a corresponding label $y \in \mathcal{Y}$. Given a set of n sample observations $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ with corresponding labels $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$, the goal of supervised learning is to learn a function $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$ that labels inputs $\boldsymbol{X}$ with their outputs $\boldsymbol{y}$ and generalizes well to unseen observations.

In **regression** tasks, the labels are assumed to be real-valued (i.e., $\mathcal{Y} = \mathbb{R}$). In **classification** tasks, the labels are a set of discrete classes; for example, in binary classification, $\mathcal{Y} = \{+1, -1\}$. We assume that the learned model for both paradigms $f_{\boldsymbol{\theta}}$ can be parameterized by a vector $\boldsymbol{\theta}$. The model is then trained to minimize the average loss over the training data between the model's predictions and the given target labels:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\big(f(\boldsymbol{x}_i, \boldsymbol{\theta}), y_i\big) + \mathcal{R}(f_{\boldsymbol{\theta}}) \ ,$$

where $\mathcal{L}(\cdot)$ is generally assumed to be a convex metric, and $\mathcal{R}(\cdot)$ regularizes the learned model. The form of the model f, loss function $\mathcal{L}(\cdot)$, and regularization method varies between learning methods. This formulation encompasses a number of parametric learning methods, such as linear regression and logistic regression to name a few.

3.2 Reinforcement Learning

A reinforcement learning (RL) agent selects sequential actions in an environment to maximize its expected return. An RL task is typically formulated as a Markov Decision Process (MDP) $\langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{X}$ is the set of states, and $\mathcal{A}$ is the set of actions that the agent may execute, $P : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is the state transition probability describing the systems dynamics, $R : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount assigned to rewards over time. At time step h, the agent is in state $\boldsymbol{x}_h \in \mathcal{X}$ and chooses an action $\boldsymbol{a} \in \mathcal{A}$ according to policy $\pi_{\boldsymbol{\theta}} : \mathcal{X} \times \mathcal{A} \mapsto [0, 1]$, which is represented as a function defined by a vector of control parameters $\boldsymbol{\theta} \in \mathbb{R}^d$. The agents then receives reward r_h according to R and transitions to state $\boldsymbol{x}_{h+1}$ according to P. This sequence of states, actions, and rewards is given as a trajectory $\boldsymbol{\tau} = \{(\boldsymbol{x}_1, \boldsymbol{a}_1, r_1), \ldots, (\boldsymbol{x}_H, \boldsymbol{a}_H, r_H)\}$ over a horizon H. The goal of RL is to find the optimal policy π^* with parameters $\boldsymbol{\theta}^*$ that

maximizes the expected cumulative discounted reward. Learning an individual task typically requires numerous trajectories, motivating the use of transfer to reduce the number of interactions with the environment.

Policy gradient (PG) methods (Sutton, McAllester, Singh, & Mansour, 1999), which we employ as our base learner for RL tasks, are a class of RL algorithms that are effective for solving high-dimensional problems with continuous state and action spaces, such as robotic control (Peters & Schaal, 2008). The goal of PG is to optimize the expected average return: $\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}\left[\frac{1}{H}\sum_{h=1}^{H} r_h\right] = \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\mathfrak{R}(\boldsymbol{\tau})d\boldsymbol{\tau}$, where $\mathbb{T}$ is the set of all possible trajectories, the average reward on trajectory $\boldsymbol{\tau}$ is given by $\mathfrak{R}(\boldsymbol{\tau}) = \frac{1}{H}\sum_{h=1}^{H} r_h$, and $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = P_0(\boldsymbol{x}_1)\prod_{h=1}^{H} p(\boldsymbol{x}_{h+1} \mid \boldsymbol{x}_h, \boldsymbol{a}_h)\ \pi(\boldsymbol{a}_h \mid \boldsymbol{x}_h)$ is the probability of $\boldsymbol{\tau}$ under an initial state distribution $P_0 : \mathcal{X} \mapsto [0, 1]$. Most PG methods, such as episodic REINFORCE (Williams, 1992), PoWER (Kober & Peters, 2009), and Natural Actor Critic (Peters & Schaal, 2008), optimize the policy by employing supervised function approximators to maximize a lower bound on the expected return of $\mathcal{J}(\boldsymbol{\theta})$. This optimization is carried out by generating trajectories using the current policy $\pi_{\boldsymbol{\theta}}$, and then comparing the result with a new policy $\pi_{\tilde{\boldsymbol{\theta}}}$. Jensen's inequality can then be used to lower bound the expected return (Kober & Peters, 2009):

$$\begin{aligned}
\log \mathcal{J}\left(\tilde{\boldsymbol{\theta}}\right) &= \log \int_{\mathbb{T}} p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})\ \mathfrak{R}(\boldsymbol{\tau})\, \mathrm{d}\boldsymbol{\tau} \\
&= \log \int_{\mathbb{T}} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}\ p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})\ \mathfrak{R}(\boldsymbol{\tau})\, \mathrm{d}\boldsymbol{\tau} \\
&\geq \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\ \mathfrak{R}(\boldsymbol{\tau}) \log \frac{p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}\, \mathrm{d}\boldsymbol{\tau} + \text{constant} \\
&\propto -\mathfrak{D}_{\mathrm{KL}}\big(p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\ \mathfrak{R}(\boldsymbol{\tau}) \mid\mid p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})\big) = \mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}\right) \quad ,
\end{aligned}$$

where $\mathfrak{D}_{\mathrm{KL}}\big(p(\boldsymbol{\tau}) \mid\mid q(\boldsymbol{\tau})\big) = \int_{\mathbb{T}} p(\boldsymbol{\tau}) \log \frac{p(\boldsymbol{\tau})}{q(\boldsymbol{\tau})}\, \mathrm{d}\boldsymbol{\tau}$. This is equivalent to minimizing the KL divergence between the reward-weighted trajectory distribution of $\pi_{\boldsymbol{\theta}}$ and the trajectory distribution $p_{\tilde{\boldsymbol{\theta}}}$ of the new policy $\pi_{\tilde{\boldsymbol{\theta}}}$.

In our work, we treat the term $\mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}\right)$ similar to the loss function $\mathcal{L}$ of a classification or regression task. Consequently, both supervised learning tasks and RL tasks can be modeled in a unified framework, where the goal is to minimize a convex loss function.

3.3 Lifelong Machine Learning

In a lifelong learning setting (Thrun, 1996; Ruvolo & Eaton, 2013; Chen & Liu, 2018), a learner faces multiple, consecutive tasks and must rapidly learn each new task by building upon its previous experience. The learner may encounter a previous task at any time, and so must optimize performance across all tasks seen so far. A priori, the agent does not know the total number of tasks $T_{\max}$, the task distribution, or the task order.

At time t, the lifelong learner encounters task $\mathcal{Z}^{(t)}$. In this paper, all tasks are either regression problems $\mathcal{Z}^{(t)} = \langle \boldsymbol{X}^{(t)}, \boldsymbol{y}^{(t)} \rangle$, classification problems $\mathcal{Z}^{(t)} = \langle \boldsymbol{X}^{(t)}, \boldsymbol{y}^{(t)} \rangle$, or reinforcement learning problems specified by an MDP $\langle \mathcal{X}^{(t)}, \mathcal{A}^{(t)}, P^{(t)}, R^{(t)}, \gamma^{(t)} \rangle$. Note that we

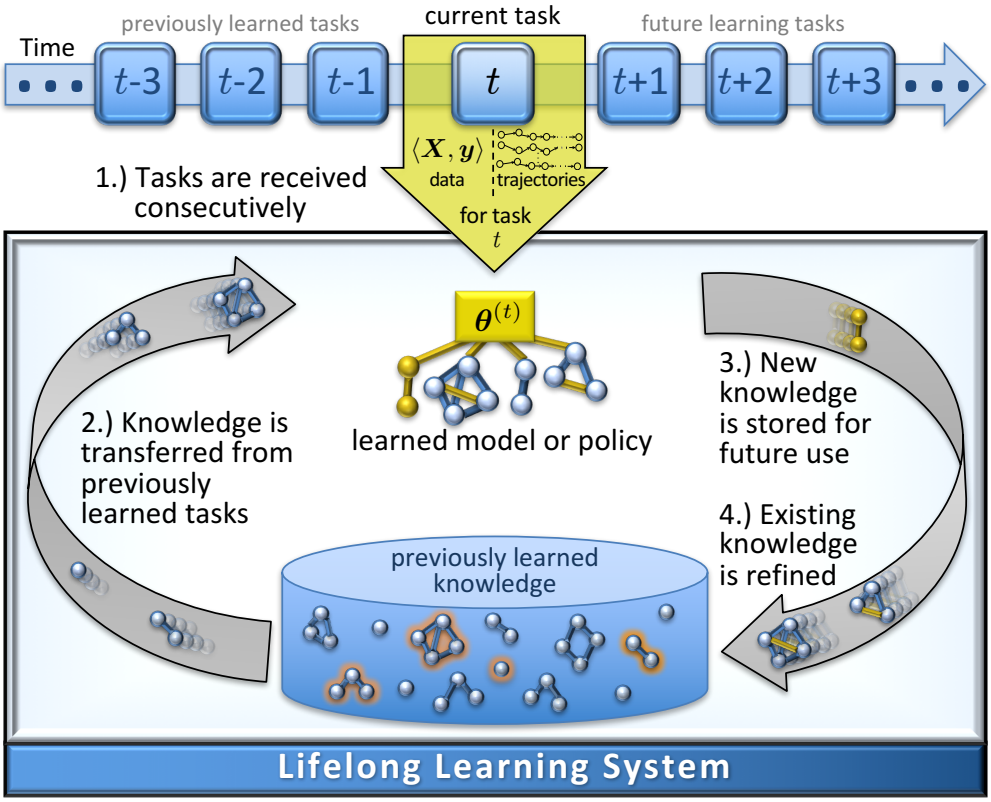


Figure 1: The lifelong machine learning process. As a new task arrives, knowledge accumulated from previous tasks is selectively transferred to the new task to improve learning. Newly learned knowledge is then stored for future use.

do not mix the learning paradigms—the lifelong learning agent will only face one type of learning task during its lifetime. The agent will learn each task consecutively, acquiring training data (i.e., samples or trajectories) in each task before advancing to the next. The agent's goal is to learn the optimal models $\{f^*_{\boldsymbol{\theta}^{(1)}}, \ldots, f^*_{\boldsymbol{\theta}^{(T)}}\}$ or policies $\{\pi^*_{\boldsymbol{\theta}^{(1)}}, \ldots, \pi^*_{\boldsymbol{\theta}^{(T)}}\}$ with corresponding parameters $\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(T)}\}$, where T is the number of unique tasks seen so far ($1 \leq T \leq T_{\text{max}}$). Ideally, knowledge learned from previous tasks $\{\mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(T-1)}\}$ should accelerate training and improve performance on each new task $\mathcal{Z}^{(T)}$. Also, the lifelong learner should scale effectively to large numbers of tasks, learning each new task rapidly from minimal data. The lifelong learning framework is depicted in Figure 1.

The Efficient Lifelong Learning Algorithm (ELLA) (Ruvolo & Eaton, 2013) and PG-ELLA (Bou Ammar et al., 2014) were developed to operate in this lifelong learning setting for classification/regression and RL tasks, respectively. Both approaches assume the parameters for each task model can be factorized using a shared knowledge base $\boldsymbol{L}$, facilitating transfer between tasks. Specifically, the model parameters for task $\mathcal{Z}^{(t)}$ are given by $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}$, where $\boldsymbol{L} \in \mathbb{R}^{d \times k}$ is the shared basis over the model or policy parameter space, and $\boldsymbol{s}^{(t)} \in \mathbb{R}^k$ are the sparse coefficients over the basis. This factorization, depicted

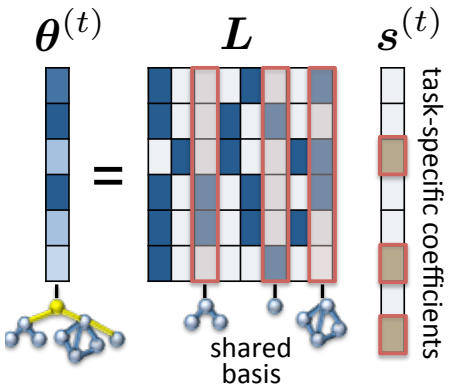


Figure 2: The task specific model (or policy) parameters $\boldsymbol{\theta}^{(t)}$ are factored into a shared knowledge repository $\boldsymbol{L}$ and a sparse code $\boldsymbol{s}^{(t)}$. The repository $\boldsymbol{L}$ stores chunks of knowledge that are useful for multiple tasks, and the sparse code $\boldsymbol{s}^{(t)}$ extracts the relevant pieces of knowledge for a particular task's model (or policy).

in Figure 2, has been effective for transfer in both lifelong and multi-task learning (Kumar & Daumé, 2012; Maurer et al., 2013).

Under this factorization, the MTL objective is:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \frac{1}{T} \sum_{t=1}^{T} \left[\mathcal{L}(\boldsymbol{\theta}^{(t)}) + \mu \|\boldsymbol{s}^{(t)}\|_1 \right] + \lambda \|\boldsymbol{L}\|_{\mathsf{F}}^2 \ , \tag{1}$$

where $\boldsymbol{S} = [\boldsymbol{s}^{(1)} \cdots \ \boldsymbol{s}^{(T)}]$ is the matrix of sparse vectors, $\mathcal{L}$ is the task-specific loss for task $\mathcal{Z}^{(t)}$, and $\|\cdot\|_{\mathsf{F}}$ is the Frobenius norm. The L_1 norm is used to approximate the true vector sparsity of $\boldsymbol{s}^{(t)}$, and μ and λ are regularization parameters. Note that for a convex loss function $\mathcal{L}(\cdot)$, this problem is convex in each of the variables $\boldsymbol{L}$ and $\boldsymbol{S}$. Thus, one can use an alternating optimization approach to solve it in a batch learning setting. To solve this objective in a lifelong learning setting, Ruvolo and Eaton (2013) take a second-order Taylor expansion to approximate the objective around an estimate $\boldsymbol{\alpha}^{(t)} \in \mathbb{R}^d$ of the single-task model parameters for each task $\mathcal{Z}^{(t)}$, and update only the coefficients $\boldsymbol{s}^{(t)}$ for the current task at each time step. This process reduces the MTL objective to the problem of sparse coding the single-task models in the shared basis $\boldsymbol{L}$, and enables $\boldsymbol{S}$ and $\boldsymbol{L}$ to be solved efficiently by the following alternating online update rules that constitute ELLA (Ruvolo & Eaton, 2013):

$$\boldsymbol{s}^{(t)} \leftarrow \arg\min_{\boldsymbol{s}} \|\boldsymbol{\alpha}^{(t)} - \boldsymbol{L}\boldsymbol{s}\|^2_{\boldsymbol{\Gamma}^{(t)}} + \mu\|\boldsymbol{s}\|_1 \tag{2}$$

$$\boldsymbol{A} \leftarrow \boldsymbol{A} + (\boldsymbol{s}^{(t)}\boldsymbol{s}^{(t)\top}) \otimes \boldsymbol{\Gamma}^{(t)} \tag{3}$$

$$\boldsymbol{b} \leftarrow \boldsymbol{b} + \text{vec}\left(\boldsymbol{s}^{(t)\top} \otimes \left(\boldsymbol{\alpha}^{(t)\top}\boldsymbol{\Gamma}^{(t)}\right)\right) \tag{4}$$

$$\boldsymbol{L} \leftarrow \text{mat}\left(\left(\frac{1}{T}\boldsymbol{A} + \lambda \boldsymbol{I}_{kd}\right)^{-1} \frac{1}{T}\boldsymbol{b}\right) \ , \tag{5}$$

where $\|\boldsymbol{v}\|_{\boldsymbol{A}}^2 = \boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v}$, the symbol $\otimes$ denotes the Kronecker product, $\boldsymbol{\Gamma}^{(t)}$ is the Hessian of the loss $\mathcal{L}(\boldsymbol{\alpha}^{(t)})$, $\boldsymbol{I}_m$ is the $m \times m$ identity matrix, $\boldsymbol{A}$ is initialized to a $kd \times kd$ zero matrix, and $\boldsymbol{b} \in \mathbb{R}^{kd}$ is initialized to zeros.

This was extended to handle reinforcement learning by Bou Ammar et al. (2014) via approximating the RL multi-task objective by first substituting in the convex lower-bound to the PG objective $\mathcal{J}(\boldsymbol{\alpha}^{(t)})$ in order to make the optimization convex.

While these methods are effective for lifelong learning, this approach requires training data to estimate the model or policy for each new task before the learner can solve it. Our key idea is to eliminate this restriction by incorporating task descriptors into lifelong learning, enabling zero-shot transfer to new tasks. That is, upon learning a few tasks, future task models or policies can be predicted solely using task descriptors.

4. Lifelong Learning with Task Descriptors

4.1 Task Descriptors

High-level descriptions not only can characterize single tasks, but can be used to identify inter-task relationships. For example, in multi-task medical domains, patients are often grouped into tasks by demographic data and disease presentation (Oyen & Lane, 2012). In control problems, the dynamical system parameters (e.g., the spring, mass, and damper constants in a spring-mass-damper system) describe the task. Descriptors can also be derived from external sources, such as text descriptions (Pennington, Socher, & Manning, 2014; Huang, Socher, Manning, & Ng, 2012) or Wikipedia text associated with the task (Socher et al., 2013).

To incorporate task descriptors into the learning procedure, we assume that each task $\mathcal{Z}^{(t)}$ has an associated descriptor $\boldsymbol{m}^{(t)}$ that is given to the learner upon first presentation of the task. The learner has no knowledge of future tasks, or the distribution of task descriptors. The descriptor is represented by a feature vector $\phi\big(\boldsymbol{m}^{(t)}\big) \in \mathbb{R}^{d_m}$, where $\phi(\cdot)$ performs feature extraction and (possibly) a non-linear basis transformation on the features. We make no assumptions on the uniqueness of $\phi\big(\boldsymbol{m}^{(t)}\big)$, although in general, we would expect different tasks to have different descriptors.[2] In addition, each task also has associated training data $\boldsymbol{X}^{(t)}$ to learn the model; in the case of RL tasks, the data consists of trajectories that are dynamically acquired by the agent through experience in the environment.

We incorporate task descriptors into lifelong learning via sparse coding with a coupled dictionary, enabling the descriptors and learned models (or policies) to augment each other. In an earlier version of this work, we focused on RL tasks (Isele et al., 2016); here, we more fully explore the range of our approach, showing how it can be applied to regression, classification, and RL problems.

4.2 Coupled Dictionary Optimization

As described previously, many multi-task and lifelong learning approaches have found success with factorizing the model or policy parameters $\boldsymbol{\theta}^{(t)}$ for each task as a sparse linear combination over a shared basis: $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}$. In effect, each column of the shared basis $\boldsymbol{L}$

[2] This raises the question of what descriptive features to use, and how task performance will change if some descriptive features are unknown. We explore these issues in Section 8.1.

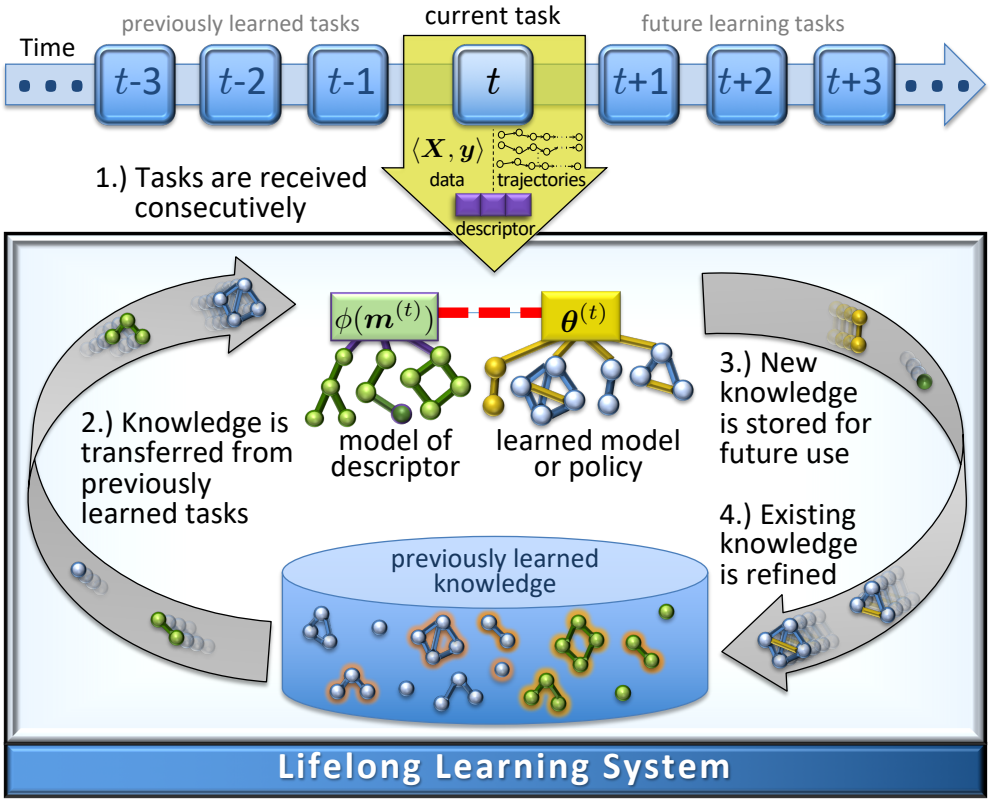


Figure 3: The lifelong machine learning process with task descriptions. High-level descriptors for each task are incorporated as input into the lifelong learning framework and coupled with the learned model or policy. Because of the learned coupling, the model or policy for a new task can be predicted given only a high-level description of that task.

serves as a reusable model or policy component representing a cohesive chunk of knowledge. In lifelong learning, the basis $\boldsymbol{L}$ is refined over time as the system learns more tasks. The coefficient vectors $\boldsymbol{S} = [\boldsymbol{s}^{(1)} \dots \boldsymbol{s}^{(T)}]$ encode the task models or policies in this shared basis, providing an embedding of the tasks based on how their models or policies share knowledge.

We make a similar assumption about the task descriptors—that the descriptor features $\phi(\boldsymbol{m}^{(t)})$ can be linearly factorized[3] using a latent basis $\boldsymbol{D} \in \mathbb{R}^{d_m \times k}$ over the descriptor space. This basis captures relationships among the descriptors, with coefficients that similarly embed tasks based on commonalities in their descriptions. From a co-view perspective (Yu, Wu, Yang, Tian, Luo, & Zhuang, 2014), both the policies and descriptors provide information about the task, and so each can augment the learning of the other. Each underlying task is common to both views, and so we seek to find task embeddings that are consistent for *both* the models/policies and their corresponding task descriptors. As depicted in Figure 4, we can enforce this by coupling the two bases $\boldsymbol{L}$ and $\boldsymbol{D}$, sharing the same coefficient vectors $\boldsymbol{S}$ to reconstruct both the models/policies and descriptors.

[3]This is potentially non-linear w.r.t $\boldsymbol{m}^{(t)}$, since ϕ can be non-linear.

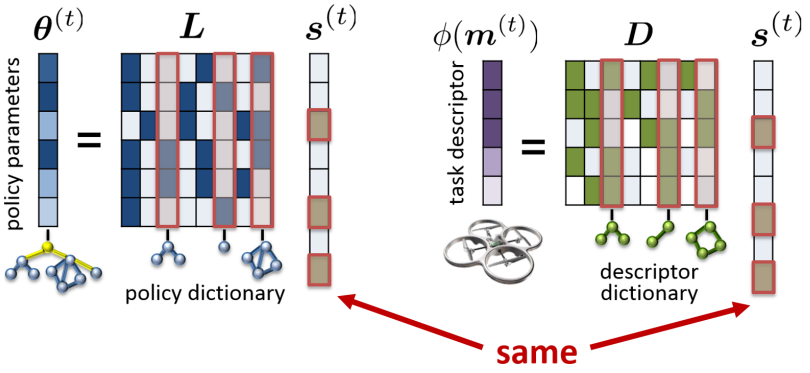


Figure 4: The coupled dictionaries of TaDeLL, illustrated on an RL task. Policy parameters $\boldsymbol{\theta}^{(t)}$ are factored into $\boldsymbol{L}$ and $\boldsymbol{s}^{(t)}$ while the task description $\phi(\boldsymbol{m}^{(t)})$ is factored into $\boldsymbol{D}$ and $\boldsymbol{s}^{(t)}$. Because we force both dictionaries to use the same sparse code $\boldsymbol{s}^{(t)}$, the relevant pieces of information for a task become coupled with the description of the task.

Therefore, for task $\mathcal{Z}^{(t)}$,

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)} \qquad \phi(\boldsymbol{m}^{(t)}) = \boldsymbol{D}\boldsymbol{s}^{(t)} \ . \tag{6}$$

To optimize the coupled bases $\boldsymbol{L}$ and $\boldsymbol{D}$ during the lifelong learning process, we employ techniques for coupled dictionary optimization from the sparse coding literature (Yang et al., 2010), which optimizes the dictionaries for multiple feature spaces that share a joint sparse representation. This notion of coupled dictionary learning has led to high performance algorithms for image super-resolution (Yang et al., 2010), allowing the reconstruction of high-res images from low-res samples, and for multi-modal retrieval (Zhuang, Wang, Wu, Zhang, & Lu, 2013) and cross-domain retrieval (Yu et al., 2014). The core idea is that features in two independent subspaces can have the same representation in a third subspace.

Given the factorization in Eq. (6), we can re-formulate the multi-task objective (Eq. (1)) for the coupled dictionaries as

$$\min_{\boldsymbol{L},\boldsymbol{D},\boldsymbol{S}} \ \frac{1}{T}\sum_{t}\left[\mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right) + \rho\left\|\phi(\boldsymbol{m}^{(t)}) - \boldsymbol{D}\boldsymbol{s}^{(t)}\right\|_2^2 + \mu\left\|\boldsymbol{s}^{(t)}\right\|_1\right] + \lambda(\|\boldsymbol{L}\|_{\mathsf{F}}^2 + \|\boldsymbol{D}\|_{\mathsf{F}}^2) \ , \tag{7}$$

where ρ balances the model's or policy's fit to the task descriptor's fit.

To solve Eq. (7) online, we approximate $\mathcal{L}(\cdot)$ by a second-order Taylor expansion around $\boldsymbol{\alpha}^{(t)}$, the ridge minimizer for the single-task learner:

$$\boldsymbol{\alpha}^{(t)} = \arg\min_{\boldsymbol{\theta}^{(t)}} \ \mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right) + \mu_s\|\boldsymbol{\theta}^{(t)}\|_2^2 \ , \tag{8}$$

where μ_s is a regularization parameter. In reinforcement learning, $\pi_{\boldsymbol{\alpha}^{(t)}}$ is the single-task policy for $\mathcal{Z}^{(t)}$ based on the observed trajectories (Bou Ammar et al., 2014). In supervised learning, $\boldsymbol{\alpha}^{(t)}$ is the single-task model parameters for $\mathcal{Z}^{(t)}$ (Ruvolo & Eaton, 2013). Note that these parameters are computed once, when the current task is learned. Then we can expand $\mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right)$ for each task around $\boldsymbol{\alpha}^{(t)}$ as:

$$\mathcal{L}\left(\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}\right) = \mathcal{L}\left(\boldsymbol{\alpha}^{(t)}\right) + \nabla\mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right)^{\top}_{\boldsymbol{\theta}^{(t)}=\boldsymbol{\alpha}^{(t)}}\left(\boldsymbol{\alpha}^{(t)} - \boldsymbol{L}\boldsymbol{s}^{(t)}\right) + \left\|\boldsymbol{\alpha}^{(t)} - \boldsymbol{L}\boldsymbol{s}^{(t)}\right\|^2_{\boldsymbol{\Gamma}^{(t)}} \ , \tag{9}$$

where ∇ denotes the gradient operator. Note that $\boldsymbol{\alpha}^{(t)}$ is the minimizer of the function $\mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right)$, and hence $\nabla\mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right)_{\boldsymbol{\theta}^{(t)}=\boldsymbol{\alpha}^{(t)}} = \mathbf{0}$. Also, since $\mathcal{L}\left(\boldsymbol{\alpha}^{(t)}\right)$ is a constant term with respect to the variables, it can be ignored for the purpose of optimization. As a result, this procedure leads to a unified and simplified formalism that is independent of the learning paradigm (i.e., classification, regression, or RL). Approximating Eq. (7) leads to

$$\min_{\boldsymbol{L},\boldsymbol{D},\boldsymbol{S}} \frac{1}{T}\sum_{t}\left[\left\|\boldsymbol{\alpha}^{(t)}-\boldsymbol{L}\boldsymbol{s}^{(t)}\right\|^2_{\boldsymbol{\Gamma}^{(t)}}+\rho\left\|\phi\big(\boldsymbol{m}^{(t)}\big)-\boldsymbol{D}\boldsymbol{s}^{(t)}\right\|_2^2+\mu\left\|\boldsymbol{s}^{(t)}\right\|_1\right]+\lambda(\|\boldsymbol{L}\|_{\mathsf{F}}^2+\|\boldsymbol{D}\|_{\mathsf{F}}^2)\ . \quad (10)$$

We can merge pairs of terms in Eq. (10) by choosing:

$$\boldsymbol{\beta}^{(t)}=\begin{bmatrix}\boldsymbol{\alpha}^{(t)}\\ \phi\big(\boldsymbol{m}^{(t)}\big)\end{bmatrix} \qquad \boldsymbol{K}=\begin{bmatrix}\boldsymbol{L}\\ \boldsymbol{D}\end{bmatrix} \qquad \boldsymbol{A}^{(t)}=\begin{bmatrix}\boldsymbol{\Gamma}^{(t)} & \mathbf{0}\\ \mathbf{0} & \rho\boldsymbol{I}_{d_m}\end{bmatrix}\ ,$$

where $\mathbf{0}$ is the zero matrix, letting us rewrite Eq. (10) concisely as

$$\min_{\boldsymbol{K},\boldsymbol{S}} \frac{1}{T}\sum_{t}\left[\left\|\boldsymbol{\beta}^{(t)}-\boldsymbol{K}\boldsymbol{s}^{(t)}\right\|^2_{\boldsymbol{A}^{(t)}}+\mu\left\|\boldsymbol{s}^{(t)}\right\|_1\right]+\lambda\|\boldsymbol{K}\|_{\mathsf{F}}^2\ . \quad (11)$$

This objective can now be solved efficiently online, as a series of per-task update rules given in Algorithm 1, which we call TaDeLL (Task Descriptors for Lifelong Learning). When a task arrives, the corresponding sparse vector $\boldsymbol{s}^{(t)}$ is computed and then the dictionaries are updated. Note that Eq. (11) can be decoupled into two optimization problems with similar form on $\boldsymbol{L}$ and $\boldsymbol{D}$, and then $\boldsymbol{L}$ and $\boldsymbol{D}$ can be updated independently using Equations 3–5, following a recursive construction based on an eigenvalue decomposition (Yu, 1991). Note that the objective function in Eq. (10) is binconvex, i.e. it is convex on each variable when the other variable is assumed fixed, and hence it can also be solved in an offline setting through alternation on the variables $\boldsymbol{K}$ and $\boldsymbol{S}$, similar to GO-MTL (Kumar & Daumé, 2012). At each iteration, one variable is fixed and the other variable is optimized in an offline setting as denoted in Algorithm 2. This gives rise to an offline version of TaDeLL which we call the TaDeMTL (Task Descriptors for Multitask Learning) algorithm. Note that TaDeMTL has a nested loop and is computationally demanding; at each iteration, sparse vectors for all tasks are recomputed and the dictionaries are updated from scratch. The major benefit is that TaDeMTL can be thought of as an upper-bound for TaDeLL, which not only can be used to assess the quality of online performance in an asymptotic regime, but also as a useful algorithm on its own for offline learning scenarios where accuracy is the priority.

For the sake of clarity, we now explicitly state the differences between using TaDeLL for RL problems and for classification and regression problems. In an RL setting, at each timestep TaDeLL receives a new RL task and samples trajectories for the new task. We use the single-task policy as computed using a twice-differentiable policy gradient method as $\boldsymbol{\alpha}^{(t)}$. The Hessian $\Gamma^{(t)}$, calculated around the point $\boldsymbol{\alpha}^{(t)}$, is derived according to the particular policy gradient method being used. Bou Ammar et al. (2014) derive it for the cases of Episodic REINFORCE and Natural Actor Critic. The reconstructed $\boldsymbol{\theta}^{(t)}$ is then used as the policy for the task $\mathcal{Z}^{(t)}$.

In the case of classification and regression, at each time step TaDeLL observes a labeled training set $(\boldsymbol{X}^{(t)}, \boldsymbol{y}^{(t)})$ for task $\mathcal{Z}^{(t)}$, where $\boldsymbol{X}^{(t)} \subseteq \mathbb{R}^{n_t \times d}$. For classification tasks, $\boldsymbol{y}^{(t)} \in$

Algorithm 1 TaDeLL (k, λ, μ)

$\boldsymbol{L} \leftarrow \text{RandomMatrix}_{d,k}$, $\boldsymbol{D} \leftarrow \text{RandomMatrix}_{m,k}$
while some task $\left(\mathcal{Z}^{(t)}, \phi\left(\boldsymbol{m}^{(t)}\right)\right)$ is available **do**
    $\mathbb{T}^{(t)} \leftarrow \text{collectData}(\mathcal{Z}^{(t)})$
    Compute $\boldsymbol{\alpha}^{(t)}$ and $\boldsymbol{\Gamma}^{(t)}$ from $\mathbb{T}^{(t)}$
    $\boldsymbol{s}^{(t)} \leftarrow \arg\min_{\boldsymbol{s}} \left\|\boldsymbol{\beta}^{(t)} - \boldsymbol{K}\boldsymbol{s}\right\|^2_{\boldsymbol{A}^{(t)}} + \mu\|\boldsymbol{s}\|_1$
    $\boldsymbol{L} \leftarrow \text{update}L(\boldsymbol{L}, \boldsymbol{s}^{(t)}, \boldsymbol{\alpha}^{(t)}, \boldsymbol{\Gamma}^{(t)}, \lambda)$ Eq. 3–5
    $\boldsymbol{D} \leftarrow \text{update}D(\boldsymbol{D}, \boldsymbol{s}^{(t)}, \phi\left(\boldsymbol{m}^{(t)}\right), \rho\boldsymbol{I}_{d_m}, \lambda)$ Eq. 3–5
    for $t \in \{1, \ldots, T\}$ **do**: $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{L}\boldsymbol{s}^{(t)}$
end while

Algorithm 2 TaDeMTL (k, λ, μ)

$\boldsymbol{L} \leftarrow \text{RandomMatrix}_{d,k}$, $\boldsymbol{D} \leftarrow \text{RandomMatrix}_{m,k}$
$\mathbb{T}^{(t)} \leftarrow \text{collectallData}(\mathcal{Z}^{(1)}, \ldots \mathcal{Z}^{(T)})$
for $itr = \{1, \ldots, N_{itr}\}$ **do**
    for $t = \{1, \ldots, T\}$ **do**
        Compute $\boldsymbol{\alpha}^{(t)}$ and $\boldsymbol{\Gamma}^{(t)}$ from $\mathbb{T}^{(t)}$
        $\boldsymbol{s}^{(t)} \leftarrow \arg\min_{\boldsymbol{s}} \left\|\boldsymbol{\beta}^{(t)} - \boldsymbol{K}\boldsymbol{s}\right\|^2_{\boldsymbol{A}^{(t)}} + \mu\|\boldsymbol{s}\|_1$
    end for
    $\boldsymbol{L} \leftarrow \text{update}L(\boldsymbol{L}, \boldsymbol{s}^{(t)}, \boldsymbol{\alpha}^{(t)}, \boldsymbol{\Gamma}^{(t)}, \lambda)$ Eq. 3–5
    $\boldsymbol{D} \leftarrow \text{update}D(\boldsymbol{D}, \boldsymbol{s}^{(t)}, \phi\left(\boldsymbol{m}^{(t)}\right), \rho\boldsymbol{I}_{d_m}, \lambda)$ Eq. 3–5
end for
for $t \in \{1, \ldots, T\}$ **do**: $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{L}\boldsymbol{s}^{(t)}$

$\{+1, -1\}^{n_t}$, and for regression tasks, $\boldsymbol{y}^{(t)} \in \mathbb{R}^{n_t}$. We then set $\boldsymbol{\alpha}^{(t)}$ to be the parameters of a single-task model trained via classification or regression (e.g., logistic or linear regression) on that data set, $\boldsymbol{\Gamma}^{(t)}$ to be the Hessian of the corresponding loss function around the single-task solution $\boldsymbol{\alpha}^{(t)}$, and the reconstructed $\boldsymbol{\theta}^{(t)}$ is used as the model parameters for the corresponding classification or regression problem.

4.3 Zero-Shot Transfer Learning

In a lifelong setting, when faced with a new task, the agent's goal is to learn an effective model or policy for that task as quickly as possible. At this stage, previous multi-task and lifelong learners incur a delay before they could produce a decent model/policy, since they need to acquire data from the new task in order to identify related knowledge and train the new model/policy via transfer.

Incorporating task descriptors enables our approach to predict a model or policy for the new task immediately, given *only* the descriptor. This ability to perform zero-shot transfer is enabled by the use of coupled dictionary learning, which allows us to observe a data instance in one feature space (i.e., the task descriptor), and then recover its underlying latent signal in the other feature space (i.e., the model or policy parameters) using the dictionaries and sparse coding.

Algorithm 3 Zero-Shot Transfer to a New Task $\mathcal{Z}^{(t_{new})}$

1: **Inputs:** task descriptor $\boldsymbol{m}^{(t_{new})}$, learned bases $\boldsymbol{L}$ and $\boldsymbol{D}$
2: $\tilde{\boldsymbol{s}}^{(t_{new})} \leftarrow \arg\min_{\boldsymbol{s}} \left\|\phi\big(\boldsymbol{m}^{(t_{new})}\big) - \boldsymbol{D}\boldsymbol{s}\right\|_2^2 + \mu \left\|\boldsymbol{s}\right\|_1$
3: $\tilde{\boldsymbol{\theta}}^{(t_{new})} \leftarrow \boldsymbol{L}\tilde{\boldsymbol{s}}^{(t_{new})}$
4: **Return:** $f_{\tilde{\boldsymbol{\theta}}^{(t_{new})}}$ or $\pi_{\tilde{\boldsymbol{\theta}}^{(t_{new})}}$

Given only the descriptor $\boldsymbol{m}^{(t_{new})}$ for a new task $\mathcal{Z}^{(t_{new})}$, we can estimate the embedding of the task in the latent descriptor space via LASSO on the learned dictionary $\boldsymbol{D}$:

$$\tilde{\boldsymbol{s}}^{(t_{new})} \leftarrow \arg\min_{\boldsymbol{s}} \left\|\phi\big(\boldsymbol{m}^{(t)}\big) - \boldsymbol{D}\boldsymbol{s}\right\|_2^2 + \mu \left\|\boldsymbol{s}\right\|_1 \quad . \tag{12}$$

Since the estimate given by $\tilde{\boldsymbol{s}}^{(t_{new})}$ also serves as the coefficients over the latent basis $\boldsymbol{L}$, we can immediately predict a model or policy for the new task as: $\tilde{\boldsymbol{\theta}}^{(t_{new})} = \boldsymbol{L}\tilde{\boldsymbol{s}}^{(t_{new})}$. This zero-shot transfer learning procedure is given as Algorithm 3.

5. Theoretical Analysis

This section examines theoretical issues related to incorporating task descriptors into lifelong learning via the coupled dictionaries. We start by proving PAC-learnability of our framework. We also outline why the inclusion of task features can improve performance of the learned models/policies and enable zero-shot transfer to new tasks safely. We then prove the convergence of TaDeLL. A full sample complexity analysis is beyond the scope of this paper, and, indeed, remains an open problem for zero-shot learning (ZSL) (Romera-Paredes & Torr, 2015).

5.1 Algorithm PAC-learnability

In this section, we establish the PAC-learnability of our algorithm. The goal is to provide bounds on the generalization error given the number of the previously learned tasks. This can help us to compute the number of required learned tasks (i.e., past experience) for the ZSL algorithm to learn future tasks from their descriptors with high probability. We rely on the ZSL framework developed by Palatucci et al. (2009). The core idea is that if we can recover the sparse vector with high accuracy through using the task descriptor, then the task parameters can also be recovered with high probability. Let P_t denote the probability of predicting the task parameters in the ZSL regime. This probability can be decomposed into two probabilities:

1. Given a certain confidence parameter δ and error parameter ϵ, a dictionary can be trained by learning $T_{\epsilon,\delta}$ previous tasks such that for future tasks $\mathbb{E}(\|\boldsymbol{\beta} - \boldsymbol{K}\boldsymbol{s}\|_2^2) \leq \epsilon$, where $\mathbb{E}(\cdot)$ denotes statistical expectation. We denote this event by $\mathcal{K}_\epsilon$, with probability $P(\mathcal{K}_\epsilon) = 1-\delta$. This event denotes that the learned knowledge has been successfully incorporated into the coupled dictionaries and we can rely on this dictionary for ZSL to succeed.

2. Given the event $\mathcal{K}_\epsilon$ (i.e., given the dictionaries learned from previous tasks), the current (future) task sparse vector can be estimated with high probability using task descriptors, enabling us to use it to compute the task parameters. We denote this event by $\mathcal{S}_\epsilon|\mathcal{K}_\epsilon$.

Since the above two events are independent, the event P_t can be expressed as the product of the above probabilities:

$$P_t = P(\mathcal{K}_\epsilon)P(\mathcal{S}_\epsilon|\mathcal{K}_\epsilon) \ . \tag{13}$$

Our goal is as follows: given the desired values for the confidence parameter δ (i.e. $P(\mathcal{K}_\epsilon) = 1-\delta$) and the error parameter ϵ (i.e. $\mathbb{E}(\|\boldsymbol{\beta}-\boldsymbol{K}\boldsymbol{s}\|_2^2) \leq \epsilon$), we compute the minimum number of tasks $T_{\epsilon,\delta}$ that needs to be learned to achieve that level of prediction confidence as well as $P(\mathcal{S}_\epsilon|\mathcal{K}_\epsilon)$ to compute P_t. To establish the error bound, we need to ensure that the coupled dictionaries are learned to a sufficient quality that achieves this error bound. We can rely on the following theorem on PAC-learnability of dictionary learning:

Theorem 5.1. *(Gribonval, Jenatton, Bach, Kleinsteuber, & Seibert, 2015) Consider the dictionary learning problem in Eq.* (11)*, the confidence parameter δ ($P(\mathcal{K}_\epsilon) = 1-\delta$), and the error parameter ϵ in the standard PAC-learning setting. Then, the number of required tasks to learn the dictionary $T_{\epsilon,\delta}$ satisfies the following relation:*

$$\begin{aligned} \epsilon &\geq 3\sqrt{\frac{\beta\log(T_{\epsilon,\delta})}{T_{\epsilon,\delta}}} + \sqrt{\frac{\beta+\log(2/\delta)/8}{T_{\epsilon,\delta}}} \\ \beta &= \frac{(d+d_m)k}{8}\max\{1,\log(6\sqrt{8}\kappa)\} \ , \end{aligned} \tag{14}$$

where κ is a constant that depends on the loss function that we use to measure the data fidelity.

Given all parameters, Eq. (14) can be solved for $T_{\epsilon,\delta}$. For example, in the asymptotic regime for learned tasks $\epsilon \propto \left(\frac{\log(T_{\epsilon,\delta})}{T_{\epsilon,\delta}}\right)^{0.5}$, and given ϵ, we can easily compute $T_{\epsilon,\delta}$.

So, according to Theorem 5.1, if we learn at least $T_{\epsilon,\delta}$ tasks to estimate the coupled dictionaries, we can achieve the required error rate ϵ. Now we need to determine the probability of recovering the task parameters in the ZSL regime, given that the learned dictionary satisfies the error bound, or $P(\mathcal{S}_\epsilon|\mathcal{K}_\epsilon)$. For this purpose, the core step in the proposed algorithm is to compute the joint sparse representation using $\boldsymbol{m}$ and $\boldsymbol{D}$. It is also important to note that Eq. (11) has a Bayesian interpretation. We can consider it a result of a maximum a posteriori (MAP) inference, where the sparse vectors are drawn from a Laplacian distribution and the coupled dictionaries are Gaussian matrices with i.i.d elements, i.e. $d_{ij} \sim \mathcal{N}(\mathbf{0}, \epsilon)$. Hence, Eq. (11) is an optimization problem that results from Bayesian inference and by solving it, we also learn a MAP estimate of the Gaussian matrix $\boldsymbol{K} = [\boldsymbol{L}, \boldsymbol{D}]^\top$. Consequently, $\boldsymbol{D}$ would be a Gaussian matrix which is used to estimate $\boldsymbol{s}$ in ZSL regime. To compute the probability of recovering the joint sparse recovery $\boldsymbol{s}$, we can rely on the following theorem for Gaussian matrices:

Theorem 5.2. *(Negahban, Yu, Wainwright, & Ravikumar, 2009) Consider the linear system $\boldsymbol{\beta} = \boldsymbol{K}\boldsymbol{s} + \boldsymbol{n}$ with a sparse solution, i.e. $\|\boldsymbol{s}\|_0 = k$, where $\boldsymbol{K} \in \mathbb{R}^{d\times k}$ is a random Gaussian matrix and $\|\boldsymbol{n}\|_2 \leq \epsilon$, i.e. $\mathbb{E}(\|\boldsymbol{\beta} - \boldsymbol{K}\boldsymbol{s}\|_2^2) \leq \epsilon$. Then the unique solution of this system can be recovered by solving an ℓ_1-minimization with probability of $(1 - e^{d\xi})$ as far as $k \leq c'd\log(\frac{k}{d})$, where c' is a constant that depends on the loss function and noise statistics, and ξ is a constant parameter.*

Theorem 5.2 suggests that in our framework, given the learned coupled dictionaries, we can recover the sparse vector with probability $P(\mathcal{S}_\epsilon|\mathcal{K}_\epsilon) = (1 - e^{(d+d_m)\xi})$ given that $k \leq c'(d_m+_d)\log(\frac{k}{d_m+d})$ for a task. This suggests that adding the task descriptors increases the probability of recovering the task parameters from $(1-e^{d\xi})$ to $(1-e^{(d+d_m)\xi})$. Moreover, we can use Eq. (12) to recover the sparse representation in the ZSL regime and subsequently unseen attributes with probability $P(\mathcal{S}_\epsilon|\mathcal{K}_\epsilon) = (1-e^{d_m\xi})$, as long as the corresponding sparse vector satisfies $k \leq c'd_m\log(\frac{k}{d_m})$ to guarantee that the recovered sparse vector is accurate enough to recover the task parameters. This theorem also suggests that the developed framework can only work if a suitable sparsifying dictionary can be learned and we have access to rich task descriptors. Therefore, given desired error $1-\delta$ and error parameter ϵ, the probability event of predicting task parameters in ZSL regime can be computed as:

$$P_t = (1-\delta)\big(1 - e^{p\xi}\big) \ , \tag{15}$$

which concludes the PAC-learnability analysis of our algorithm. Given the learnability of our approach, the next question is whether the proposed dictionary learning algorithm computationally converges to a suitable solution.

5.2 Theoretical Convergence of TaDeLL

In this section, we prove the convergence of TaDeLL, showing that the learned dictionaries become increasingly stable as it learns more tasks. We build upon the theoretical results from Bou Ammar et al. (2014) and Ruvolo & Eaton (2013), demonstrating that these results apply to coupled dictionary learning with task descriptors, and use them to prove convergence.

Let $\hat{g}_T(\mathrm{L})$ represent the sparse-coded approximation to the MTL objective, which can be defined as:

$$\hat{g}_T(\boldsymbol{L}) = \frac{1}{T}\sum_{t=1}^{T} \|\boldsymbol{\alpha}^{(t)} - \boldsymbol{L}\boldsymbol{s}^{(t)}\|^2_{\boldsymbol{\Gamma}^{(t)}} + \mu\|\boldsymbol{s}^{(t)}\|_1 + \lambda\|\boldsymbol{L}\|_{\mathsf{F}}^2 \ .$$

This equation can be viewed as the cost for $\boldsymbol{L}$ when the sparse coefficients are kept constant. Let $\boldsymbol{L}_T$ be the version of the dictionary $\boldsymbol{L}$ obtained after observing T tasks. Given these definitions, we consider the following theorem:

Theorem 5.3. *(Ruvolo & Eaton, 2013)*

1. *The trained dictionary $\boldsymbol{L}$ is stabilized over learning with rate: $\boldsymbol{L}_T - \boldsymbol{L}_{T-1} = O(\frac{1}{T})$*
2. *$\hat{g}_T(\boldsymbol{L}_T)$ converges almost surely.*
3. *$\hat{g}_T(\boldsymbol{L}_T) - \hat{g}_T(\boldsymbol{L}_{T-1})$ converges almost surely to zero.*

This theorem requires two conditions:

1. The tuples $\mathbf{\Gamma}^{(t)}$, $\boldsymbol{\alpha}^{(t)}$ are drawn i.i.d from a distribution with compact support to bound the norms of $\boldsymbol{L}$ and $\boldsymbol{s}^{(t)}$.

2. For all t, let $\boldsymbol{L}_\kappa$ be the subset of the dictionary $\boldsymbol{L}_t$, where only columns corresponding to non-zero element of $\boldsymbol{s}^{(t)}$ are included. Then, all eigenvalues of the matrix $\boldsymbol{L}_\kappa^\mathsf{T}\mathbf{\Gamma}^{(t)}\boldsymbol{L}_\kappa$ need to be strictly positive.

Bou Ammar et al. (2014) show that both of these conditions are met for the lifelong learning framework given in Eqs. 2–5. When we incorporate the task descriptors into this framework, we alter $\boldsymbol{\alpha}^{(t)} \rightarrow \boldsymbol{\beta}^{(t)}$, $\boldsymbol{L} \rightarrow \boldsymbol{K}$, and $\mathbf{\Gamma}^{(t)} \rightarrow \boldsymbol{A}^{(t)}$. Note both $\boldsymbol{\beta}^{(t)}$ and $\boldsymbol{A}^{(t)}$ are formed by adding deterministic entries and thus can be considered to be drawn i.i.d (because $\mathbf{\Gamma}^{(t)}$ and $\boldsymbol{\alpha}^{(t)}$ are assumed to be drawn i.i.d). Therefore, incorporating task descriptors does not violate Condition 1.

To show that Condition 2 holds, if we analogously form $\boldsymbol{K}_\kappa$, then the eigenvalues of $\boldsymbol{K}_\kappa$ are strictly positive because they are either eigenvalues of $\boldsymbol{L}$ (which are strictly positive according to Bou Ammar et al., 2014) or the regularizing parameter ρ by definition. Thus, both conditions are met and convergence follows directly from Theorem 5.3.

5.3 Computational Complexity

In this section, we analyze the computational complexity of TaDeLL. Each update begins with updating $\boldsymbol{\alpha}^{(t)}$ and $\mathbf{\Gamma}^{(t)}$ at a cost of $O(\xi(d, n_t))$, where $\xi(\cdot)$ depends on the base PG learner for RL and the based supervised learning method for regression/classification and n_t is the number of trajectories for RL and the number of data points for regression/classification, obtained for task $\mathcal{Z}^{(t)}$. The cost of updating $\boldsymbol{L} \in \mathbb{R}^{d\times k}$ and $\boldsymbol{s}^{(t)} \in \mathbb{R}^k$ alone is $O(k^2d^3)$ (Ruvolo & Eaton, 2013), and so the cost of updating $\boldsymbol{K} \in \mathbb{R}^{(d+d_m)\times k}$ through coupled dictionary learning is $O(k^2(d+d_m)^3)$. This yields an overall per-update cost of $O(k^2(d+d_m)^3 + \xi(d, n_t))$, which is independent of T.

Next, we empirically demonstrate the benefits of TaDeLL on a variety of different learning problems.

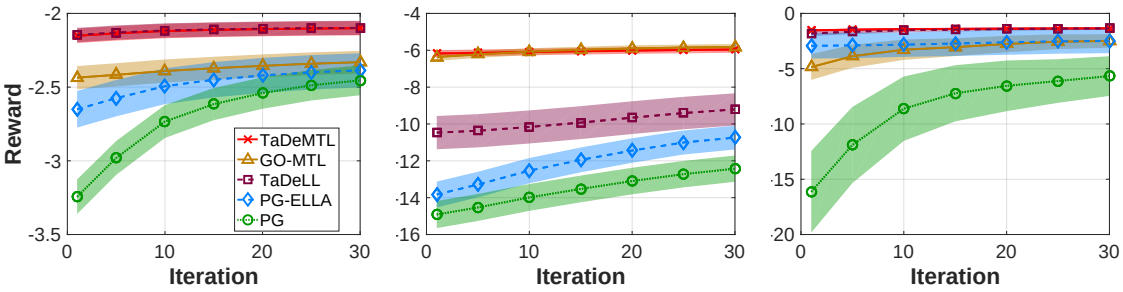
6. Evaluation on Reinforcement Learning Domains

We applied TaDeLL to a series of RL problems, where the challenge is to learn a collection of different, related systems. For these systems, we use three benchmark control problems and an application to quadrotor stabilization.

6.1 Benchmark Dynamical Systems

Spring Mass Damper (SM) The SM system is commonly used for its ability to represent oscillations. It is described by three parameters: the spring constant, mass, and damping constant. The system's state is given by the position and velocity of the mass. The controller applies a force to the mass, attempting to stabilize it to a given position.

Cart Pole (CP) The CP system involves balancing an inverted pendulum by applying a force to the cart. The system is characterized by the cart and pole masses, pole length, and



(a) Simple Mass (b) Cart Pole (c) Bicycle

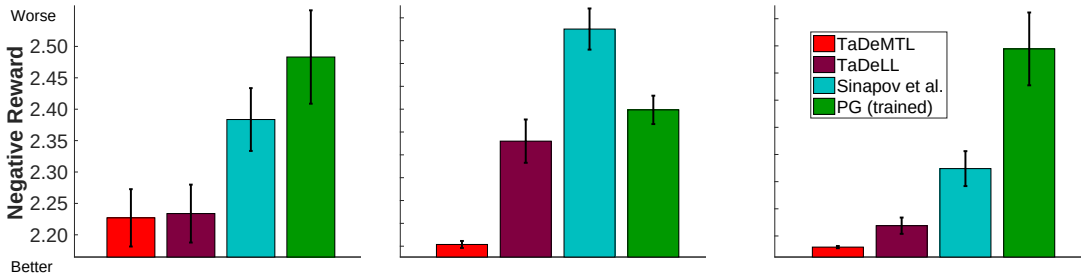
Figure 5: Performance of multi-task (solid lines), lifelong (dashed), and single-task learning (dotted) on benchmark dynamical systems. (Best viewed in color.)

a damping parameter. The states are the position and velocity of the cart and the angle and rotational velocity of the pole.

Bicycle (BK) This system focuses on keeping a bicycle balanced upright as it rolls along a horizontal plane at constant velocity (see subsection 6.4.2 in Busoniu, 2010). The system is characterized by the bicycle mass, x- and z-coordinates of the center of mass, and parameters relating to the shape of the bike (the wheelbase, trail, and head angle). The state is the bike's tilt and its derivative; the actions are the torque applied to the handlebar and its derivative.

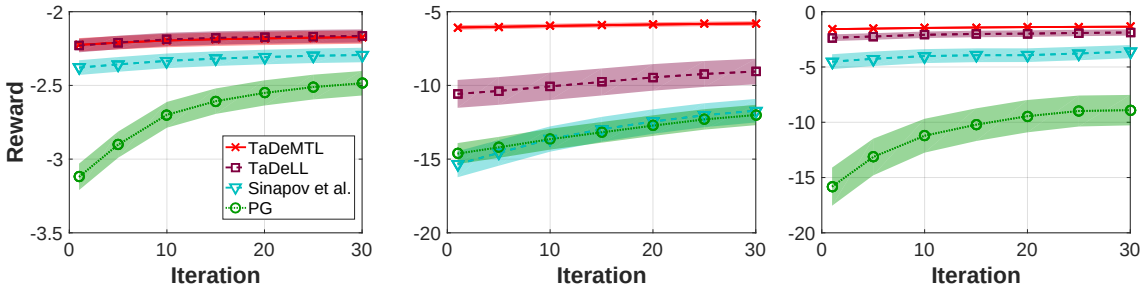
6.2 Methodology

In each domain we generated 40 tasks, each with different dynamics, by varying the system parameters. To this end, we set a maximum value and a minimum value for each task parameter and then generated the systems by uniformly drawing values for the parameters from each parameter range. The reward for each task was taken to be the distance between the current state and the goal. For lifelong learning, tasks were encountered consecutively with repetition, and learning proceeded until each task had been seen at least once. In order to cancel out the effect of task order, we ran each experiment 100 times and reported the average performance and standard deviation error. In each experiment, we used the same random task order between methods to ensure fair comparison. The learners sampled trajectories of 100 steps, and the learning session during each task presentation was limited to 30 iterations. For MTL, all tasks were presented simultaneously. We used Natural Actor Critic (Peters & Schaal, 2008) as the base learner for the benchmark systems and episodic REINFORCE (Williams, 1992) for quadrotor control. We chose k and the regularization parameters independently for each domain and the GO-MTL, ELLA, and PG-ELLA methods, selecting parameter values that optimize the combined performance of all methods on 20 held-out tasks by using a grid search over ranges $\{10^{-n} \mid n = 0, \ldots, 3\}$ for regularization parameters and $\{1, \ldots, 10\}$ for k, respectively. We set $\rho = \text{mean}(\text{diag}(\rho^{(t)}))$ to balance the fit to the descriptors and the policies. We measured learning curves based on the final policies for each of the 40 tasks. The system parameters for each task were used as the task descriptors $\phi(\boldsymbol{m})$; we also tried several non-linear transformations as $\phi(\cdot)$, but found the linear features worked well.



(a) Simple Mass (b) Cart Pole (c) Bicycle

Figure 6: Zero-shot transfer to new tasks. The plots show the initial "jumpstart" improvement on each task domain. (Best viewed in color.)



(a) Simple Mass (b) Cart Pole (c) Bicycle

Figure 7: Learning performance of using the zero-shot policies as warm start initializations for PG. The performance of the single-task PG learner is included for comparison. (Best viewed in color.)

6.3 Results on Benchmark Systems

Figure 5 compares our TaDeLL approach for lifelong learning with task descriptors to

1. PG-ELLA (Bou Ammar et al., 2014), which does not use task features,
2. GO-MTL (Kumar & Daumé, 2012), the MTL optimization of Eq. (1), and
3. single-task learning using PG.

For comparison, we also performed an offline MTL optimization of Eq. (7) via alternating optimization, and plot the results as TaDeMTL. The shaded regions on the plots denote standard error bars.

We see that task descriptors improve lifelong learning on every system, even driving performance to a level that is unachievable from training the policies from experience alone via GO-MTL in the SM and BK domains. The difference between TaDeLL and TaDeMTL is also negligible for all domains except CP, demonstrating the effectiveness of our online optimization.

To measure zero-shot performance, we generated an additional 40 tasks for each domain, averaging results over these new tasks. We compared our work mainly against Sinapov et al. (2015)'s method by using task descriptors as "task features" in that work. To make Sinapov et al. (2015)'s method applicable in a lifelong learning setting, we used their method to transfer knowledge from the tasks that has been learned before time t at each time step using a version of their method that uses linear regression to select the source

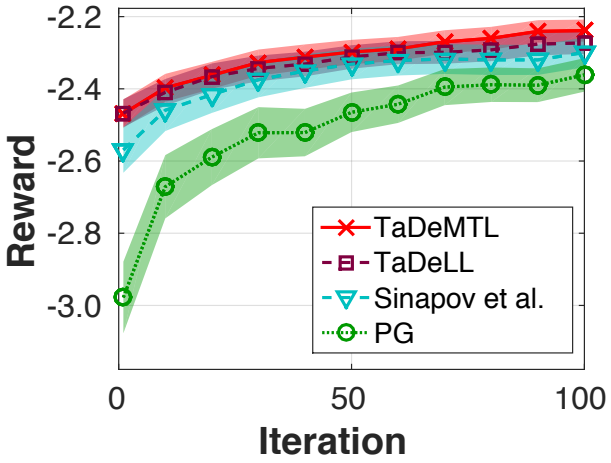


Figure 8: Warm start learning on quadrotor control. (Best viewed in color.)

task. In order to use this method in the lifelong learning setting, we need a memory buffer to save the value functions for the previously learned tasks. However, learning can still occur in an online setting by transferring knowledge from all previously learned tasks by using the suitable value function as the initial policy. Figure 6 shows that task descriptors are effective for zero-shot transfer to new tasks. We see that our approach improves the initial "jumpstart" performance (Taylor & Stone, 2009) on new tasks, outperforming Sinapov et al. (2015)'s method and single-task PG, which was allowed to train on the task. We attribute the especially poor performance of Sinapov et al. on CP to the fact that the CP policies differ substantially; in domains where the source policies are vastly different from the target policies, Sinapov et al.'s algorithm does not have an appropriate source to transfer. Their approach is also much more computationally expensive (quadratic in the number of tasks) than our approach (linear in the number of tasks), as shown in Figure 14; details of the runtime experiments are included in Section 8.2. Figure 7 shows that the zero-shot policies can be used effectively as a warm start initialization for a PG learner, which is then allowed to improve the policy.

6.4 Application to Quadrotor Control

We also applied our approach to the more challenging domain of quadrotor control, focusing on zero-shot transfer to new stability tasks. To ensure realistic dynamics, we use the model of Bouabdallah and Siegwart (2005), which has been verified on physical systems. The quadrotors are characterized by three inertial constants and the arm length, with their state consisting of roll/pitch/yaw and their derivatives.

Figure 8 shows the results of our application, demonstrating that TaDeLL can predict a controller for new quadrotors through zero-shot learning that has equivalent accuracy to PG, which had to train on the system. As with the benchmarks, TaDeLL is effective for warm start learning with PG.

Algorithm	Lifelong Learning	Zero-Shot Prediction
TaDeLL	**0.131** $\pm$ 0.004	0.159 $\pm$ 0.005
ELLA	0.152 $\pm$ 0.005	N/A
STL	0.730 $\pm$ 0.007	N/A

Table 1: Regression performance on robot end effector prediction in both lifelong learning and zero-shot settings. Performance is measured in mean-squared-error.

7. Evaluation on Supervised Learning Domains

In this section, we evaluate TaDeLL on regression and classification domains, considering the problem of predicting the real-valued location of a robot's end effector and two synthetic classification tasks.

7.1 Predicting the Location of a Robot End Effector

In this section, we evaluate TaDeLL on a regression domain. We look at the problem of predicting the real-valued position of the end effector of an 8-DOF robotic arm in 3D space, given the angles of the robot joints. Different robots have different link lengths, offsets, and twists, and we use these parameters as the description of the task.

We consider 200 different robot arms and use 10 points as training data per robot. The robot arms are simulated using the Robot Toolbox (Corke, 2011). The learned dictionaries are then used to predict models for 200 different unseen robots. We measure performance as the mean square error of the prediction against the true location of the end effector.

Table 1 shows that both TaDeLL and ELLA outperform the single-task learner, with TaDeLL slightly outperforming ELLA as the result of transfer learning. Moreover, this same improvement holds for zero-shot prediction on new robot arms. To compare the performance of TaDeLL for zero-shot prediction, we computed the single-task learner performance on the new robot using the data, which turned out to be 0.70 ± 0.05. We use STL as a baseline to measure zero-shot prediction quality using our method. STL performance in Table 1 demonstrates that TaDeLL outperforms STL on new tasks, despite not using data.

To better understand the relationship of dictionary size to performance, we investigated how learning performance varies with the number of bases k in the dictionary. Figure 10 shows this relationship for the lifelong learning and zero-shot prediction settings. We observe that TaDeLL performs better with a larger dictionary than ELLA. We hypothesize that this difference results from encoding the task descriptions. When the dimension of the input vector increases, a bigger dictionary can more easily encode the input as a representation vector. To test this hypothesis, we reduced the number of descriptors in an ablative experiment. Recall that the task has 24 descriptors consisting of a twist, link offset, and link length for each joint. We reduced the number of descriptors by alternatingly removing the subsets of features corresponding to the twist, offset, and length. Figure 11 shows the performance of this ablative experiment, revealing that the need for the increased number of bases is particularly related to learning *twist*.

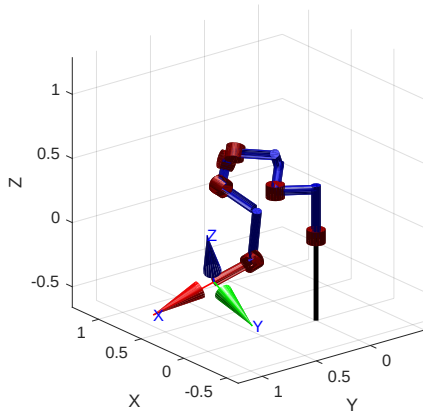
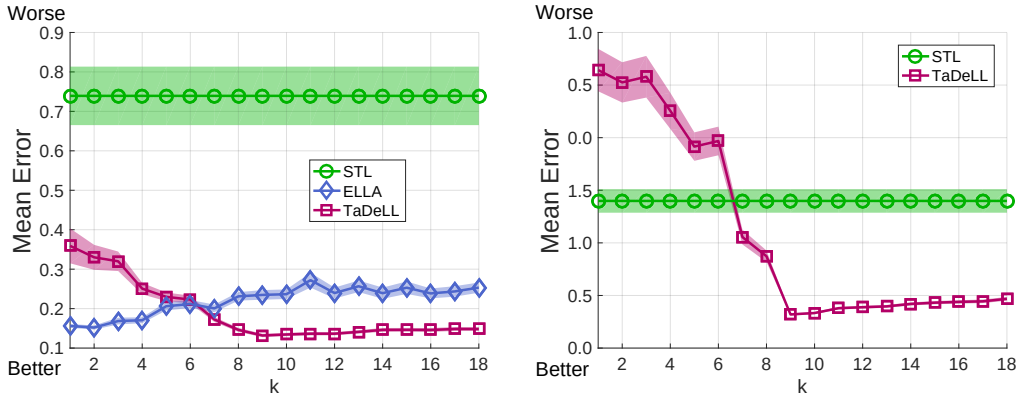


Figure 9: Example model of an 8-DOF robot arm.



(a) Lifelong Learning.

(b) Zero-shot Prediction.

Figure 10: Performance of TaDeLL and ELLA as the dictionary size k is varied for lifelong learning and zero-shot learning. Performance of the single task learner is provided for comparison. In the lifelong learning setting, both TaDeLL and ELLA demonstrate positive transfer that converges to the performance of the single-task learner as k is increased. We see that, for this problem, TaDeLL prefers a slightly larger value of k.

7.2 Experiments on Synthetic Classification Domains

To better understand the connections between TaDeLL's performance and the structure of the tasks, we evaluated TaDeLL on two synthetic classification domains. The use of synthetic domains allows us to tightly control the task generation process and the relationship between the target model and the descriptor.

The first synthetic domain consists of binary-labeled instances drawn from $\mathbb{R}^8$, and each sample $\boldsymbol{x}$ belongs to the positive class iff $\boldsymbol{x}^\mathsf{T}\boldsymbol{m} > 0$. Each task has a different parameter vector $\boldsymbol{m}$ drawn from the uniform distribution $\boldsymbol{m} \in [-0.5, 0.5]$; these vectors $\boldsymbol{m}$ are also used as the task descriptors. Note that by sampling $\boldsymbol{m}$ from the uniform distribution, this domain violates the assumptions of ELLA that the samples are drawn from a common set

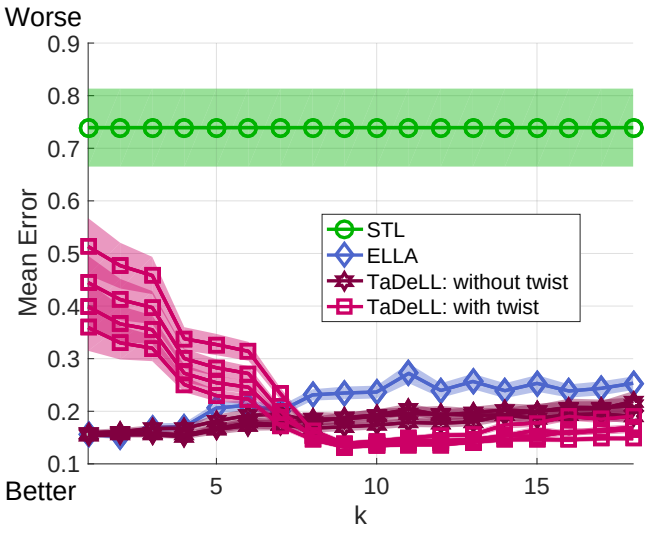


Figure 11: An ablative experiment studying the performance of TaDeLL as a function of the dictionary size k, as we vary the subset of descriptors used. The feature consist of twist(t), length(l), and offset(o) variables for each joint. We trained TaDeLL using only subsets of the features $\{t, l, o, tl, to, lo, tlo\}$ and we see that the need for a larger k is directly related to learning the *twist*. Subsets that contain twist descriptors are shown in magenta. Trials that do not include twist descriptors are shown in gray. Performance of ELLA and the single-task learner (STL) are provided for comparison. (Best viewed in color.)

Algorithm	Lifelong Learning	Zero-Shot Prediction
TaDeLL	**0.926** $\pm$ 0.004	0.930 $\pm$ 0.002
ELLA	0.814 $\pm$ 0.008	N/A
STL	0.755 $\pm$ 0.009	N/A

Table 2: Classification accuracy on Synthetic Domain 1.

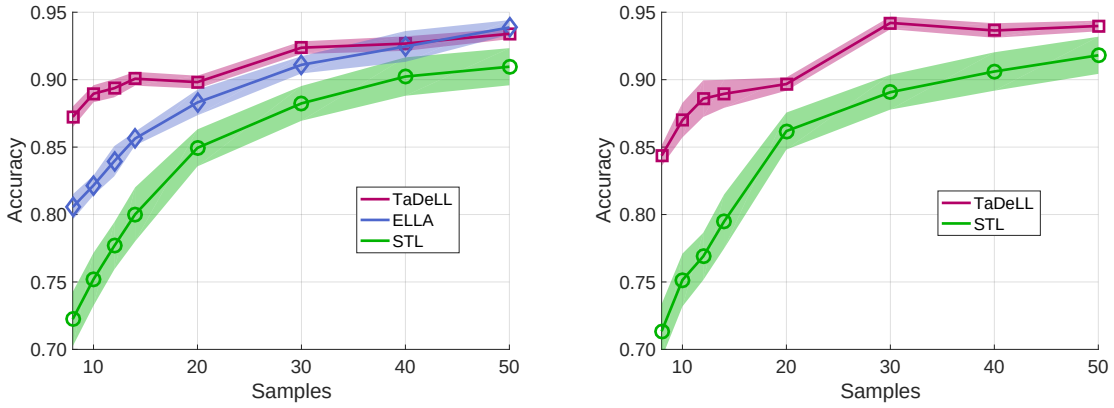
of latent features. Each task's data consists of 10 training samples, and we generated 100 tasks to evaluate lifelong learning.

Table 2 shows the performance on this Synthetic Domain 1. We see that the inclusion of meaningful task descriptors enables TaDeLL to learn a better dictionary than ELLA in a lifelong learning setting. We also generated an additional 100 unseen tasks to evaluate zero-shot prediction, which is similarly successful.

For the second synthetic domain, we generated $\boldsymbol{L}$ and $\boldsymbol{D}$ matrices, and then generated a random sparse vector $\boldsymbol{s}^{(t)}$ for each task. The true task model is then given by a logistic regression classifier with $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}$. This generation process directly follows the assumptions of ELLA and TaDeLL, where $\boldsymbol{D}$ is generated independently. We similarly generate 100 tasks for lifelong learning and another 100 unseen tasks for zero-shot prediction, and use the true task models to label 10 training points per task. In this experiment, we empirically demonstrate that TaDeLL works in the case of this assumption (Table 3) in both lifelong

Algorithm	Lifelong Learning	Zero-Shot Prediction
TaDeLL	**0.889** ± 0.006	0.87 ± 0.01
ELLA	0.821 ± 0.007	N/A
STL	0.752 ± 0.009	N/A

Table 3: Classification accuracy on Synthetic Domain 2.



(a) Lifelong Learning

(b) Zero-Shot Prediction

Figure 12: Performance versus sample complexity on Synthetic Domain 2.

learning and zero-shot prediction settings. For comparison, the baseline STL performance using data is equal to 0.762 ± 0.008 and 0.751 ± 0.009, respectively for these two settings.

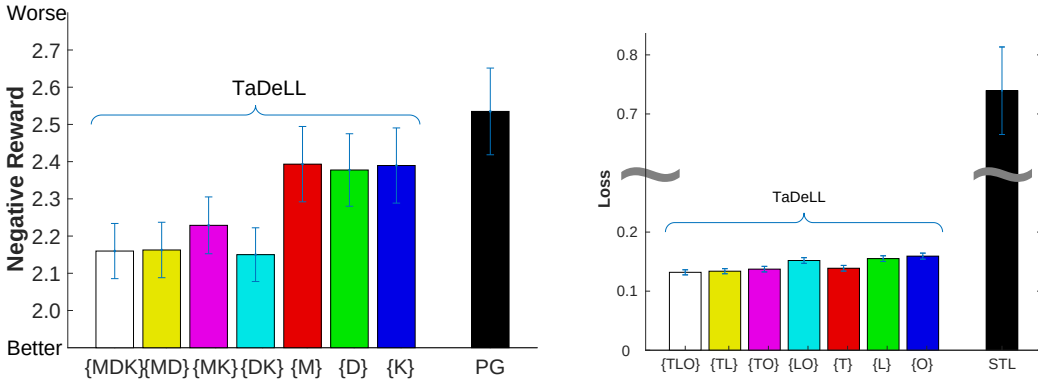
We also use this domain to investigate performance versus sample complexity, as we generated varying amounts of training data per task. In Figure 12a, we see that TaDeLL is able to greatly improve performance given only a small number of samples, and as expected, its benefit becomes less dramatic as the single-task learner receives sufficient samples. Figure 12b shows similar behavior in the zero-shot case.

8. Additional Experiments

Having shown how TaDeLL can improve learning in a variety of settings, we now turn our attention to understanding other aspects of the algorithm. Specifically, we look at the issue of task descriptor selection and partial information, runtime comparisons, and the effect of varying the number of tasks used to train the dictionaries.

8.1 Choice of Task Descriptor Features

For RL, we used the system parameters as the task description, and for the robot end effector prediction, we used the dimensions of the robot. While in these cases the choice of task descriptor was straightforward, this might not always be the case. It is unclear exactly



(a) Spring-Mass RL

(b) Robot End Effector Prediction

Figure 13: Performance using various subsets of the SM system parameters (mass M, damping constant D, and spring constant K) and Robot system parameters (twist T, link length L, and offset O) as the task descriptors.

how the choice of task descriptor features might affect the resulting performance. In other scenarios, we may have only partial knowledge of the system parameters.

To address these questions, we conducted additional experiments on the Spring-Mass (SM) system and robot end effector problem, using various subsets of the task descriptor features when learning the coupled dictionaries. Figure 13a shows how the number and selection of parameters affects performance on the SM domain. We evaluated jumpstart performance when using all possible subsets of the system parameters as the task descriptor features. These subsets of the SM system parameters (mass M, damping constant D, and spring constant K) are shown along the horizontal axis for the task descriptors. Overall, the results show that the learner performs better when using larger subsets of the system parameters as the task descriptors.

The robot task has 24 descriptors consisting of a twist, link offset, and link length for each joint. We group the subset of features describing twist, offset, and length together and examine removing different subsets. Figure 13b show that twist is more important than the other features and again the inclusion of more features improves performance.

8.2 Computational Efficiency

We compared the average per-task runtime of our approach to that of Sinapov et al. (2015), the most closely related method to our approach. Since Sinapov et al.'s method requires training transferability predictors between all pairs of tasks, its total runtime grows quadratically with the number of tasks. In comparison, our online algorithm is highly efficient. As shown in Section 5.3, the per-update cost of TaDeLL is $O\left(k^2(d+m)^3 + \xi(d, n_t)\right)$. Note that this per-update cost is independent of the number of tasks T, giving TaDeLL a total runtime that scales linearly in the number of tasks.

Figure 14 shows the per-task runtime for each algorithm based on a set of 40 tasks, as evaluated on an Intel Core i7-4700HQ CPU. TaDeLL samples tasks randomly with replacement and terminates once every task has been seen. For Sinapov et al., we used ten PG

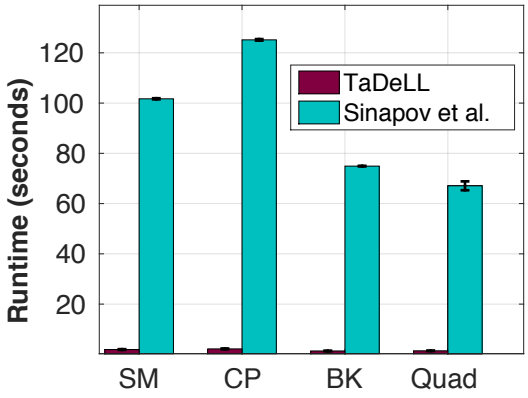


Figure 14: Runtime comparison.

iterations for calculating the warm start, ensuring fair comparison between the methods. These results show a substantial reduction in computational time for TaDeLL: two orders of magnitude over the 40 tasks.

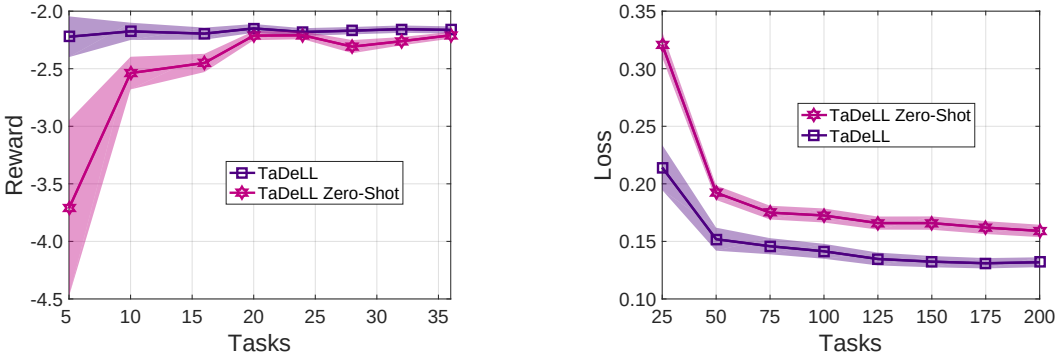
8.3 Performance for Various Numbers of Tasks

Although we have shown in Section 5.2 that the learned dictionaries become more stable as the system learns more tasks, we cannot currently guarantee that this will improve the performance of zero-shot transfer. To evaluate the effect of the number of tasks on zero-shot performance, we conducted an additional set of experiments on both the Spring-Mass domain and the robot end effector prediction domain. Our results, shown in Figure 15, reveal that zero-shot performance does indeed improve as the dictionaries are trained over more tasks. This improvement is most stable and rapid in an MTL setting, since the optimization over all dictionaries and task policies is run to convergence, but TaDeLL also shows clear improvement in zero-shot performance as T_{max} increases. Since zero-shot transfer involves only the learned coupled dictionaries, we can conclude that the quality of these dictionaries for zero-shot transfer improves as the system learns more tasks.

9. Conclusion

This article demonstrated that incorporating high-level task descriptors into lifelong learning both improves learning performance and also enables zero-shot transfer to new tasks. The mechanism of using a coupled dictionary to connect the task descriptors with the learned models or policies is relatively straightforward, yet highly effective in practice. Most critically, it provides a fast and simple mechanism to predict the model or policy for a new task via zero-shot learning, given only its high-level task descriptor. This approach is general and can handle multiple learning paradigms, including classification, regression, and RL tasks. Experiments demonstrate that our approach outperforms other lifelong learning methods and requires substantially less computational time than competing methods.

The ability to rapidly bootstrap models (or policies) for new tasks is critical to the development of lifelong learning systems that will be deployed for extended periods in real



(a) Spring-Mass RL

(b) Robot End Effector Prediction

Figure 15: Zero-shot performance as a function of the number of tasks used to train the dictionary. As more tasks are used, the performance of zero-shot transfer improves.

environments and tasked with handling a variety of tasks. High-level descriptions provide an effective way for humans to communicate and to instruct one another. The description need not come from another agent; humans often read instructions and then complete a novel task quite effectively. Enabling lifelong learning systems to similarly take advantage of these high-level descriptions provides an effective step toward their practical effectiveness. As shown in our experiments with warm-start learning from the zero-shot predicted policy, these task descriptors can also be combined with training data on the new task in a hybrid approach. Also, while our framework is designed to work for tasks that are drawn from a single domain, an interesting potential direction for the future is to extend this work for cross-domain tasks, e.g. balancing tasks of bicycle and spring-mass systems together.

Despite TaDeLL's strong performance, defining what constitutes an effective task descriptor for a group of related tasks remains an open question. In our framework, task descriptors are given, typically as fundamental descriptions of the system. The representation we use for the task descriptors, a feature vector, is also relatively simple. One interesting direction for future work is to develop methods for integrating more complex task descriptors into MTL or lifelong learning. These more sophisticated mechanism could include natural language descriptions, step-by-step instructions, or logical relationships. Such an advance would likely involve moving beyond the linear framework used in TaDeLL, but would constitute an important step toward enabling more practical use of high-level task descriptors in lifelong learning.

Acknowledgments

This research was supported by ONR grant #N00014-11-1-0139, AFRL grant #FA8750-14-1-0069, AFRL grant #FA8750-16-1-0109, and the DARPA Lifelong Learning Machines program under grant #FA8750-18-2-0117. We would like to thank the anonymous reviewers of the conference version of this paper as well as the current version for their helpful feedback.

References

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, *6*, 1817–1853.

Bakker, B., & Heskes, T. (2003). Task clustering and gating for Bayesian multitask learning. *The Journal of Machine Learning Research*, *4*, 83–99.

Baxter, J. (2000). A model of inductive bias learning. *The Journal of Artificial Intelligence Research*, *12*, 149–198.

Bickel, S., Sawade, C., & Scheffer, T. (2009). Transfer learning by distribution matching for targeted advertising. *Advances in Neural Information Processing Systems*, 145–152.

Bonilla, E. V., Agakov, F. V., & Williams, C. (2007). Kernel multi-task learning using task-specific features. *In Proceedings of the International Conference on Artificial Intelligence and Statistics*, 43–50.

Bou Ammar, H., Eaton, E., & Ruvolo, P. (2014). Online multi-task learning for policy gradient methods. *In Proceedings of the International Conference on Machine Learning*.

Bouabdallah, S., & Siegwart, R. (2005). Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. *In Proceedings of the 2005 IEEE International Conference on Robotics and Automation.*, 2247–2252.

Caruana, R. (1997). Multitask Learning. *Machine Learning*, *28*, 41–75.

Cavallanti, G., Cesa-Bianchi, N., & Gentile, C. (2010). Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, *11*, 2901–2934.

Chen, Z., & Liu, B. (2018). Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *12*(3), 1–207.

Corke, P. I. (2011). Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer.

Da Silva, F. L., & Costa, A. H. R. (2017). Towards zero-shot autonomous inter-task mapping through object-oriented task description. In *Workshop on Transfer in Reinforcement Learning (TiRL)*.

Dekel, O., Long, P. M., & Singer, Y. (2006). Online multitask learning. In *Proceedings of the International Conference on Computational Learning Theory*, pp. 453–467. Springer.

Evgeniou, T., & Pontil, M. (2004). Regularized multi–task learning. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 109–117. ACM.

Gribonval, R., Jenatton, R., Bach, F., Kleinsteuber, M., & Seibert, M. (2015). Sample complexity of dictionary learning and other matrix factorizations. *IEEE Transactions on Information Theory*, *61*(6), 3469–3486.

Ham, J., Lee, D. D., & Saul, L. K. (2005). Semisupervised alignment of manifolds. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pp. 120–127.

Hao, N., Oghbaee, A., Rostami, M., Derbinsky, N., & Bento, J. (2016). Testing fine-grained parallelism for the admm on a factor-graph. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 835–844. IEEE.

Huang, E. H., Socher, R., Manning, C. D., & Ng, A. (2012). Improving word representations via global context and multiple word prototypes. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, 873–882.

Isele, D., Rostami, M., & Eaton, E. (2016). Using task features for zero-shot knowledge transfer in lifelong learning. *In Proceedings of the International Joint Conference on Artificial Intelligence*.

Kober, J., & Peters, J. (2009). Policy search for motor primitives in robotics. *Advances in Neural Information Processing Systems*, 849–856.

Kolouri, S., Rostami, M., Owechko, Y., & Kim, K. (2018). Joint dictionaries for zero-shot learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kumar, A., & Daumé, H. (2012). Learning task grouping and overlap in multi-task learning. *In Proceedings of the International Conference on Machine Learning*, 1383–1390.

Lazaric, A., & Ghavamzadeh, M. (2010). Bayesian multi-task reinforcement learning. In *Proceedings of International Conference on Machine Learning*, pp. 599–606. Omnipress.

Long, M., Zhu, H., Wang, J., & Jordan, M. I. (2017). Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2208–2217. JMLR. org.

Maurer, A., Pontil, M., & Romera-Paredes, B. (2013). Sparse coding for multitask and transfer learning. *In Proceedings of the International Conference on Machine Learning*, *28*, 343–351.

Negahban, S., Yu, B., Wainwright, M., & Ravikumar, P. (2009). A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Advances in Neural Information Processing Systems*, 1348–1356.

Oyen, D., & Lane, T. (2012). Leveraging domain knowledge in multitask Bayesian network structure learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Palatucci, M., Hinton, G., Pomerleau, D., & Mitchell, T. M. (2009). Zero-shot learning with semantic output codes. *Advances in Neural Information Processing Systems*.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, *12*, 1532–1543.

Peters, J., & Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, *71*(7), 1180–1190.

Romera-Paredes, B., & Torr, P. H. S. (2015). An embarrassingly simple approach to zero-shot learning. *Proceedings of International Conference on Machine Learning*, 2152–2161.

Rostami, M., Kolouri, S., Eaton, E., & Kim, K. (2019). Deep transfer learning for few-shot sar image classification. *Remote Sensing*, *11*(11), 1374.

Rostami, M., Kolouri, S., Kim, K., & Eaton, E. (2018). Multi-agent distributed lifelong learning for collective knowledge acquisition. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 712–720. International Foundation for Autonomous Agents and Multiagent Systems.

Rostami, M., Kolouri, S., McClelland, J., & Pilly, P. (2020). Generative continual concept learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. AAAI Press.

Rostami, M., Kolouri, S., & Pilly, P. K. (2019). Complementary learning for overcoming catastrophic forgetting using experience replay. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 3339–3345. AAAI Press.

Ruvolo, P., & Eaton, E. (2013). ELLA: An efficient lifelong learning algorithm. In *Proceedings of the International Conference on Machine Learning*, pp. 507–515.

Saha, A., Rai, P., Venkatasubramanian, S., & Daume, H. (2011). Online learning of multiple tasks and their relationships. *Proceedings of International Conference on Artificial Intelligence and Statistics*, 643–651.

Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320.

Silva, F. L. D., & Costa, A. H. R. (2018). Object-oriented curriculum generation for reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1026–1034.

Sinapov, J., Narvekar, S., Leonetti, M., & Stone, P. (2015). Learning inter-task transferability in the absence of target task samples. *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*.

Socher, R., Ganjoo, M., Manning, C. D., & Ng, A. Y. (2013). Zero-shot learning through cross-modal transfer. *Advances in Neural Information Processing Systems*, 935–943.

Song, J., Gao, Y., & Wang, H. (2018). Feature learning and transfer performance prediction for video reinforcement learning tasks via a siamese convolutional neural network. In *International Conference on Neural Information Processing*, pp. 350–361. Springer.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, *99*, 1057–1063.

Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., & Stone, P. (2017). Automatic curriculum graph generation for reinforcement learning agents.. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2590–2596.

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, *10*, 1633–1685.

Taylor, M. E., Stone, P., & Liu, Y. (2007). Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, *8*, 2125–2167.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first?. *Advances in Neural Information Processing Systems*, 640–646.

Wang, C., & Mahadevan, S. (2009). A general framework for manifold alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3-4), 229–256.

Wilson, A., Fern, A., Ray, S., & Tadepalli, P. (2007). Multi-task reinforcement learning: a hierarchical Bayesian approach. In *Proceedings of the International Conference on Machine Learning*, pp. 1015–1022. ACM.

Xu, X., Hospedales, T. M., & Gong, S. (2016). Multi-task zero-shot action recognition with prioritised data augmentation. In *Proceedings of the European Conference on Computer Vision*, pp. 343–359. Springer.

Yang, J., Wright, J., Huang, T. S., & Ma, Y. (2010). Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, *19*(11), 2861–2873.

Yu, K. (1991). Recursive updating the eigenvalue decomposition of a covariance matrix. *IEEE Transactions on Signal Processing*, *39*(5), 1136–1145.

Yu, Z., Wu, F., Yang, Y., Tian, Q., Luo, J., & Zhuang, Y. (2014). Discriminative coupled dictionary hashing for fast cross-media retrieval. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 395–404. ACM.

Zhong, L. W., & Kwok, J. T. (2012). Convex multitask learning with flexible task clusters. *Proceedings of the International Conference on Machine Learning*, 49–56.

Zhuang, Y. T., Wang, Y. F., Wu, F., Zhang, Y., & Lu, W. M. (2013). Supervised coupled dictionary learning with group structures for multi-modal retrieval. *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*.